Утвержден

25.00152-01 97 06-ЛУ

ОПЕРАЦИОННАЯ СИСТЕМА МОС ВП

Подсистема системного сервиса

Программы системного обслуживания

Справочный материал

25.00152-01 97 06

Листов 1195/5К

1987

Перв.примен.

25.00152-01.

Литера

АННОТАЦИЯ

Настоящий документ предназначен для программистов разрабатывающих программы на языках программирования МАКРОАССЕМБЛЕР, БЕЙСИК, КОБОЛ, ФОРТРАН, ПАСКАЛЬ (в дальнейшем языках МАКРОАССЕМБЛЕР, БЕЙСИК, КОБОЛ, ФОРТРАН, ПАСКАЛЬ).

В документе представлена информация о работе и использовании каждой программы системного обслуживания.

В разделе 1 приведены общие сведения о программах системного обслуживания.

В разделе 2 описаны форматы вызовов программ системного обслуживания.

В разделе 3 приведена информация о программах системного обслуживания по обеспечению защиты.

В разделе 4 приведена информация о служебных программах обслуживания флагов событий.

В разделе 5 приведена информация о служебных программах обработки асинхронных системных прерываний.

В разделе 6 приведена информация о служебных программах обслуживания логических имен.

В разделе 7 приведена информация о служебных программах обслуживания ввода-вывода.

В разделе 8 приведена информация о программах системного обслуживания управления процессами.

В разделе 9 приведена информация о программах системного обслуживания таймера и преобразования времени.

В разделе 10 приведена информация о служебных программах обработки кода состояния.

В разделе 11 приведена информация о программах системного обслуживания управления памятью.

В разделе 12 приведена информация о программах системного обслуживания управления захватом.

В разделе 13 описаны все программы системного обслуживания.

СОДЕРЖАНИЕ

1. Введение в программы системного обслуживания

Операционная система МОС ВП использует программы системного обслуживания для управления ресурсами, доступными процессам, обеспечения связи между процессами и выполнения основных функций операционной системы, таких как координация операций ввода-вывода.

Хотя большинство программ системного обслуживания используются главным образом самой операционной системой для обслуживания подключенных пользователей, однако многие из этих программ доступны для общего использования и предоставляют механизм, который может быть использован в прикладных программах. Например, когда пользователь подключается к операционной системе МОС ВП, вызывается программа системного обслуживания ¤CREPRC (создать процесс), которая создает процесс для этого пользователя. Пользователь, в свою очередь, может написать программу, которая вызывает программу системного обслуживания ¤CREPRC, чтобы создать процесс для выполнения определенных функций.

Программы системного обслуживания можно разделить на функциональные группы. В табл.1 перечислены все группы программ системного обслуживания и их функции.

Таблица 1

Группа служебных программ	Функция
Защита	Служебные программы группы "защита" обеспечивают различные механизмы, которые может использовать пользователь для повышения уровня защиты операционной системы МОС ЭП
Флаг события	Процесс может использовать флаги событий для синхронизации последовательностей операций в программе. Служебные программы группы "флаг события" сбрасывают, устанавливают и считывают флаги событий, а также помещают процессы в состояние ожидания, откладывая установку флага или флагов событий
AST	Выполнение процесса может прерываться различными событиями (такими как завершение ввода-вывода). По такому прерыванию выполняются определенные подпрограммы. Эти программные прерывания называются асинхронными системными прерываниями (AST - это аббревиатура соответствующего английского названия), поскольку они происходят асинхронно по от-

Продолжение табл. 1

Группа служебных программ	Функция
	ношению к выполнению процесса. Программы системного обслуживания группы "AST" предназначены для управления работой AST-прерываний
Логические имена	Служебные программы группы "логические имена" предоставляют обобщенную технику для поддержки и доступа к символьной строке, называемой логическим именем, а также для установления эквивалентности пары имен. Концепция логических имен обеспечивает для системы и прикладных программ независимость операций ввода-вывода от устройств
Ввод-вывод	Служебные программы группы "ввод-вывод" выполняют операции ввода-вывода непосредственно, не используя служебные программы работы с файлами, входящие в систему управления данными (СУД). служебные программы группы "ввод-вывод" предназначены для: 1) выполнения логических, физических и виртуальных операций ввода-вывода;

Продолжение табл. 1

Группа служебных программ	Функция
	2) форматирования выводных строк с преобразованием двоичных чисел в строки кода КОИ-8 и подстановкой переменных данных в строки кода КОИ-8; 3) выполнение операций с сетями; 4) постановки сообщений в очередь к системным процессам
Управление процессами	Служебные программы группы "управление процессами", предоставляют возможность создавать, удалять и управлять процессами
Таймер и преобразование времени	Служебные программы таймера планируют программные события на конкретное время дня или по истечении заданного интервала времени. Служебные программы преобразования времени дают возможность получать и формировать двоичные значения времени для использования в служебных программах таймера
Обработка исключительных состояний	Программы обработки исключительных состояний - это программы, которые получают управление при возникновении исключи-

Продолжение табл. 1

Группа служебных программ	Функция
	тельного состояния во время выполнения образа. Служебные программы этой группы используются для специальных целей
Управление памятью	Служебные программы управления памятью дают возможность сделать доступным программе виртуальное адресное пространство. В эту группу входят служебные программы, которые могут: 1) дать возможность образу увеличить или изменить размер доступной виртуальной памяти; 2) управлять страничным обменом и обменом рабочих наборов в виртуальной памяти; 3) создавать и организовывать доступ к файлам в памяти, которые содержат разделяемый код или данные
Изменение режима	Служебные программы группы "изменение режима" меняют режим доступа процесса на более привилегированный режим для выполнения отдельных программ, или меняют указатель стека для менее привилегиро-

Продолжение табл. 1

Группа служебных программ	Функция
	ванного режима. Эти служебные программы используются главным образом операционной системой МОС ЭП
Управление захватом	Служебные программы управления захватом позволяют взаимосвязанным процессам синхронизировать свой доступ к разделяемым ресурсам

1.1. Формат документирования программ системного обслуживания

Каждая программа системного обслуживания документируется с использованием структурного формата, называемого шаблоном программы. В данном подразделе описаны основные заголовки шаблона программы, информация, представляемая под каждым заголовком, а также форматы представления информации.

Цель данного подраздела - об'яснить, где найти информацию и как правильно ее прочитать, но не как правильно ее использовать.

Некоторые основные заголовки в шаблоне программы содержат информацию, которая не требует дополнительных пояснений, кроме приведенных в табл. 2. Однако для следую-

щих основных заголовков, содержащих информацию, требуется дополнительное описание, которое приводится в остальных подразделах данного раздела:

1) заголовок формата;

2) заголовок аргументов;

3) заголовок возвращаемых значений кодов состояния.

Таблица 2

Основные заголовки в шаблоне программы

Основной заголовок	Описание
Имя программы	Требуется. Имя точки входа в программу
Коммента-рий программы	Требуется. Комментарий находится непосредственно под именем программы. Комментарий об'ясняет обычно одним-двумя предложениями, что делает программа
Формат	Требуется. Заголовок формата следует за описанием программы. Формат дает имя точки входа программы и список аргументов
Аргументы	Требуется. Заголовок аргументов следует за заголовком формат. Под этим заголовком сообщается подробная информация о каждом аргументе.
Описание	Не обязательно. Заголовок описания следует за заголовком аргументов. Текст описания содержит информацию о специфических действиях, выполня-

Продолжение табл. 2

Основной заголовок	Описание
	емых программой; взаимодействие между аргументами программы; работа программы в контексте операционной системы МОС ВП; привилегии пользователя, которые требуются для вызова программы; системные ресурсы, используемые программой и пользовательские квоты, которые могут влиять на работу программы.
	для некоторых простых программ текст описания не требуется, поскольку вся необходимая информация содержится в комментарии программы
Возвращаемые значения кодов состояния	Требуется. Заголовок возвращаемых значений кодов состояния следует за заголовком описания. Под этим заголовком перечисляются все значени кодов состояния (обычно это или коды завершения, или коды состояния), которые возвращает программа

1.1.1. Заголовок "формат"

Под заголовком "формат" представлены следующие два типа информации:

1) формат вызова программы;

2) пояснительный текст.

Все программы системного обслуживания имеют формат вызова процедуры. Использование формата вызова процедуры приводит к вызову программы в соответствии со стандартом вызова процедур и обработки исключительных состояний операционной системы МОС ВП, т.е. создается маска входа, сохраняются регистры и т.д.

Пояснительный текст располагается после формата вызова программы. Данный текст присутствует, только если необходимо пояснить формат (или форматы). Например, формат вызова указывает, что какие-то аргументы не обязательны, заключая их в квадратные скобки. Однако, сами по себе квадратные скобки не могут передать всю важную информацию, которая может относиться к необязательным аргументам. Например, в некоторых программах, которые могут иметь необязательные аргументы, если применяется один необязательный аргумент, то и другие необязательные аргументы должны быть использованы. Этот факт и поясняет текст, следующий за форматом вызова.

В примере приводится формат вызова программы, который записывается после заголовка формата. Хотя и предполагается, что здесь приводится формат в самом общем виде, на самом деле он до некоторой степени специфичен и выбран здесь для того, чтобы выявить некоторые синтаксические механизмы, которые используются для оформления более сложных вызовов процедур.

Пример.

ENTRY_POINT_NAME ARG1, ARG2, [ARG3], NULLARG [,ARG4][,ARG5]

Данный формат служит примером использования синтаксических правил приведенных в табл. 3.

Таблица 3

Элемент	Синтаксическое правило
Пробелы	Один или более пробелов должны присутствовать между именем точки входа и первым аргументом, а также между аргументами
Квадратные скобки.	В квадратных скобках записываются необязательные аргументы: ARG3, ARG4 и ARG5 являются необязательными аргументами, поскольку они записаны в квадратных скобках. запятые тоже могут быть необязательными (элемент "запятая")
Запятые	Между аргументами запятая всегда следует за пробелом. Если аргумент необязательный, то запятая может появиться либо внутри, либо вне квадратных скобок, в зависимости от позиции аргумента в списке и от того, являются ли окружающие аргументы обязательными или нет. Например, в приведенном примере аргумент ARG3 является необязательным, но

Продолжение табл. 3

Элемент	Синтаксическое правило
	поскольку в списке есть другие обязательные аргументы, следующие за ARG3, то запятая обязательна (т.к. она отмечает место ARG3), поэтому запятая расположена вне квадратных скобок. Аргументы ARG4 и ARG5 не обязательны. Поскольку в списке нет обязательных аргументов, следующих за ARG4 и ARG5, то и сами запятые перед ARG4 и ARG5 являются необязательными. Это означает, что если в вызове не заданы ARG4 и ARG5, то не будут указаны и эти запятые. Поэтому запятые перед ARG4 и ARG5 записаны внутри квадратных скобок
Нулевые аргументы	Нулевой аргумент - это аргумент, занимающий место. Он используется, чтобы удерживать место в списке аргументов. При вызове программы, содержащей нулевой аргумент, необходимо либо обеспечить значение 0 для нулевого аргумента, либо не давать ему никакого значения, но отметить запятой его место в формате вызова

1.1.2. Заголовок "аргументы"

Под заголовком "аргументы" приводится подробная информация о каждом аргументе, перечисленном в формате вызова. Аргументы описываются в том порядке, в котором они расположены в формате вызова.

Для определения каждого аргумента используются следующие элементы:

Имя-аргумента

Использование в МОС ВП:

Тип:

Доступ:

Механизм:

1.1.2.1. Элемент "использование в операционной системе МОС ВП"

Каждый тип данных операционной системы МОС ВП имеет только одно представление в памяти, например, тип данных МОС ВП "ACCESS_MODE" (тип доступа) представляет из себя байт без знака. Кроме того, тип данных МОС ВП может иметь или не иметь "определенного" значения.

Большинство типов данных МОС ВП можно рассматривать как "определенные", т.е. они несут значение, которое является уникальным в контексте операционной системы мос вл. Например, тип данных МОС ВП "ACCESS_MODE". В памяти он представлен одним байтом без знака, а определенное содержимое этого байта без знака связано с тем фактом, что он ука-

зывает режим аппаратного доступа и имеет, таким образом, только четыре допустимых значения:

0 - режим ядра;

1 - режим управления;

2 - режим супервизора;

3 - пользовательский режим.

Однако, некоторые типы данных МОС ВП не являются определенными, то есть они задают представление в памяти, но не несут другого семантического содержания с точки зрения операционной системы МОС ВП, например, тип данных МОС ВП "BYTE_SIGNED" (байт со знаком) не является определенным.

В МОС ВП используются следующие типы данных:

ACCESS_BIT_NAMES -

однородный массив из 32 дескрипторов, каждый длиной в квадрослово. Каждый дескриптор определяет имя одного из 32 битов в маске доступа. Первый дескриптор именует бит 0, второй дескриптор именует бит 1 и т.д;

ACCESS_MODE (режим доступа) -

байт без знака, означающий режим аппаратного доступа. Байт без знака может иметь четыре значения:

0 - режим ядра;

1 - режим управления;

2 - режим супервизора;

3 - пользовательский режим.

ADDRESS (адрес) -

длинное слово без знака, означающее адрес в виртуальной памяти данных или кода, но не маски входа в проце-

дуру (который относится к типу "PROCEDURE");

ADDRESS_RANGE (диапазон адресов) -

квадрослово без знака, определяющее диапазон адресов в виртуальной памяти, который задает область памяти. первое длинное слово указывает чачальный адрес в диапазоне, а второе длинное слово задает конечный адрес в диапазоне;

ARG_LIST -

список аргументов процедуры, состоящий из длинных слов, число которых может меняться от 1 до 256. Первое длинное слово содержит целое число без знака, равное количеству следующих за ним последовательных длинных слов, каждое из которых представляет аргумент, который должен передаваться процедуре посредством инструкции CALL. Формат списка аргументов приведен на рис. 1.

Формат списка аргументов

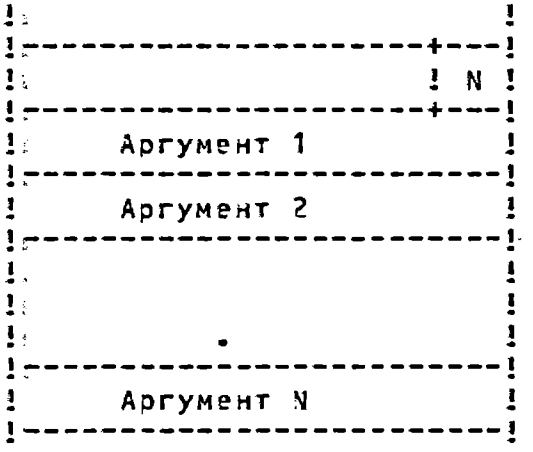


Рис. 1

AST_PROCEDURE (AST-процедура) -

длинное слово без знака, представляющее маску входа в процедуру, которая должна вызываться на уровне AST. (Процедуры, которые не должны вызываться на уровне AST относятся к типу "PROCEDURE");

BOOLEAN (булев) -

длинное слово без знака, обозначающее булев флаг с двумя значениями:

0 - ложь;

1 - истина;

BYTE_SIGNED (байт со знаком) -

тип данных МОС ВП "байт со знаком" совпадает с типом данных "целое число (со знаком) размером в байт", представленном в табл. 5;

BYTE_UNSIGNED (байт без знака) -

тип данных МОС ВП "байт без знака" совпадает с типом данных "байт (без знака)", представленном в табл. 5;

CHANNEL (канал) -

целое число без знака длиной в слово, которое является индексом канала ввода-вывода;

CHAR_STRING (строка символов) -

строка длиной от 0 до 65535 символов. Этот тип данных МОС ВП совпадает с типом данных "строка символов", представленном в табл. 5. На рис. 2 показана строка символов "XYZ".

Строка символов

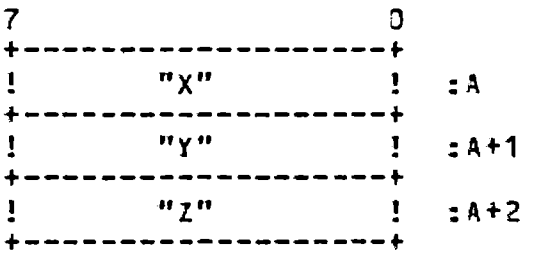


Рис. 2

COMPLEX_NUMBER (комплексное число) -

один из стандартных типов данных операционной системы МОС ВП представляющих комплексное число с плавающей запятой. Их может быть три:

1) комплексное число с плавающей запятой F-формата;

2) комплексное число с плавающей запятой D-формата;

3) комплексное число с плавающей запятой G-формата.

Комплексное число с плавающей запятой F-формата (R,I) состоит из двух чисел с плавающей запятой F-формата. Первое число с плавающей запятой F-формата есть действительная часть (R) комплексного числа, второе число с плавающей запятой F-формата есть мнимая часть (I). Структура комплексного числа с плавающей запятой F-формата приведена на рис. 3.

Комплексное число с плавающей запятой F-формата

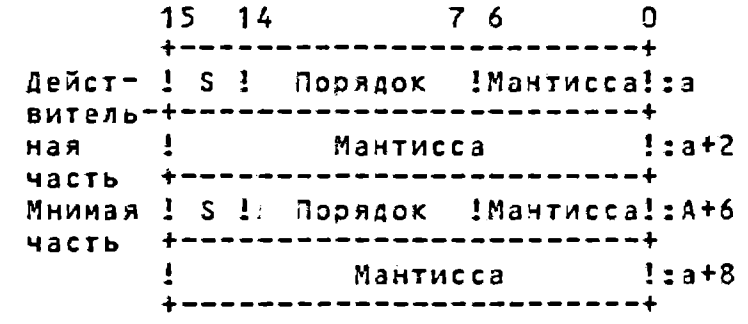


Рис. 3

Комплексное число с плавающей запятой D-формата (R, I) состоит из двух чисел с плавающей запятой D-формата. Первое число с плавающей запятой D-формата есть действительная часть (R) комплексного числа, второе число с плавающей запятой D-формата есть мнимая часть (I) комплексного числа. Структура комплексного числа с плавающей запятой D-формата приведена на рис. 4

Комплексное число с плавающей запятой D-формата

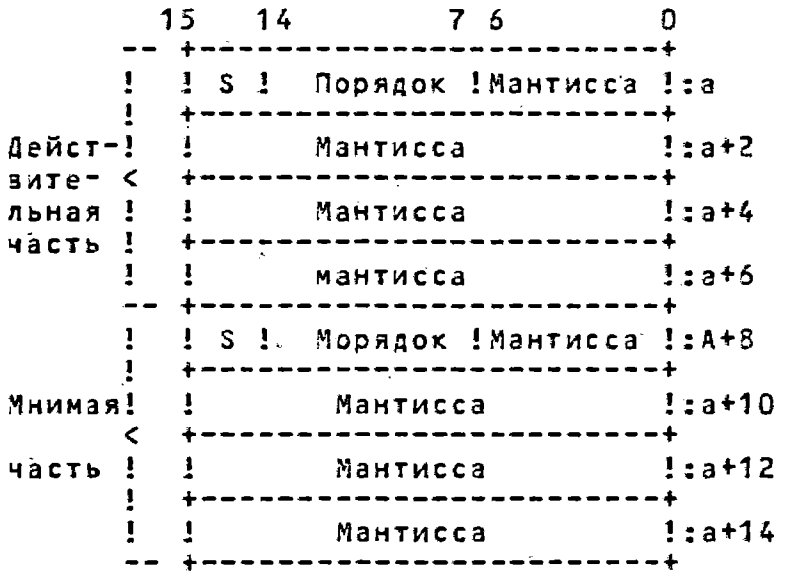


Рис. 4

Комплексное число с плавающей запятой G-формата (R, I) состоит из двух чисел с плавающей запятой G-формата. Первое число с плавающей запятой G-формата есть действительная часть (R) комплексного числа, второе число с плавающей запятой G-формата есть мнимая часть (I) комплексного числа. Структура комплексного числа с плавающей запятой G-формата приведена на рис. 5.

Комплексное число с плавающей запятой G-формата

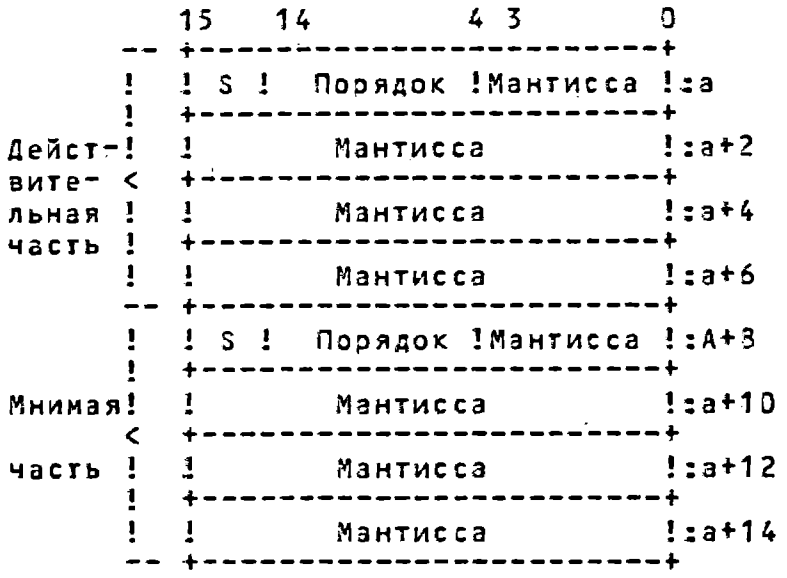


Рис. 5

COND_VALUE -

Целое число размером в длинное слово, определяющее значение кода состояния (т.е. состояние возврата или системный код состояния), которое обычно возвращается процедурой в регистре R0. Значение кода состояния имеет структуру, изображенную на рис. 6. В зависимости от конкретных нужд пользователя он может проверять либо только самый младший бит, либо три младших бита, либо все значение целиком. Самый младший бит указывает на успешное (1) или неуспешное (0) завершение обслуживания. Три младших бита, взятые вместе, отражают серьезность ошибки. Оставшиеся биты (с 3 по 31) классифицируют конкретное состояние возврата и компоненту операционной системы МОС ВП, выдавшую код состояния. Каждое числовое значение кода состояния имеет уникаль-

ное символическое имя, формат которого SS¤CODE - код, где код- это мнемоническое имя, описывающее состояние возврата.

Формат кода состояния

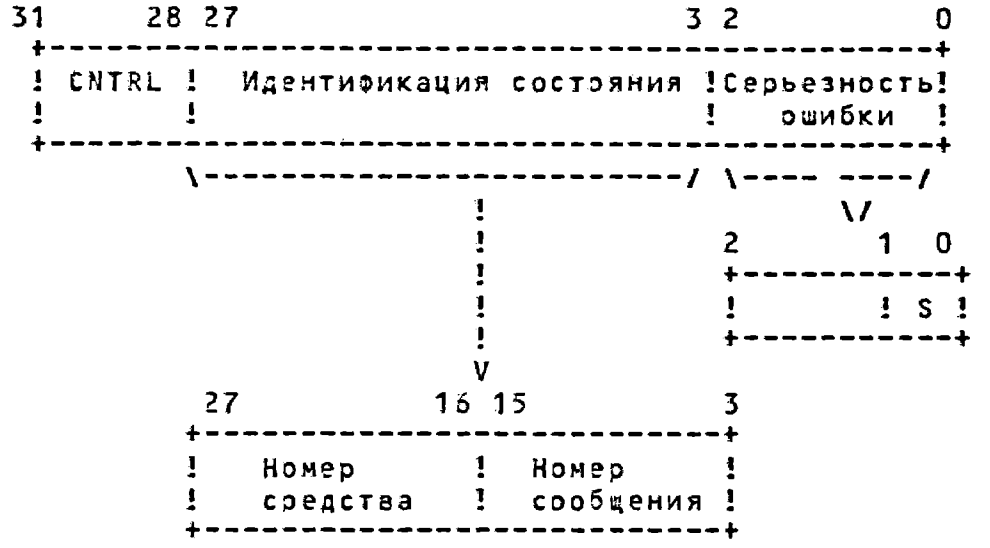


Рис. 6

CONTEXT (контекст) -

длинное слово без знака, которое используется вызываемой процедурой для управления положением в итеративной последовательности вызовов. Оно обычно инициализируется в вызывающем об'екте, но после этого управляется вызванной программой;

DATA-TIME (дата-время) -

целое двоичное число без знака размером в 64 бита, обозначающее дату и время в виде количества 100-наносекундных интервалов, прошедших с момента 00 часов 00 мин. 17 ноября 1858 года. Тип данных МОС ВП "дата-время" совпадает с типом данных "абсолютная дата

и время", представленном в табл. 5;

DEVICE-NAME -

строка символов, обозначающая имя устройства. Длина строки может составлять от одного до 15 символов. Это может быть логическое имя, но в таком случае оно должно транслироваться в допустимое имя устройства. Если имя устройства содержит двоеточие (:), то это двоеточие и все символы после него игнорируются. Если строке с именем устройства предшествует символ подчеркивания (_), это указывает на то, что строка является физическим именем устройства;

EF_CLUSTER_NAME (имя кластера флагов событий) -

строка символов, обозначающая имя кластера флагов событий. Длина может составлять от 1 до 15 символов. Это может быть логическое имя, но в таком случае оно должно транслироваться в допустимое имя кластера флагов событий (раздел 4);

EF_NUMBER (номер флага события) -

целое число без знака размером в длинное слово, обозначающее номер флага события. Для программ пользователей доступны флаги, пронумерованные от 32 до 63;

EXIT_HANDLER_BLOCK (блок программы управления выходом) -

структура переменной длины, задающая блок управления для программы управления выходом. Управляющий блок, который описывает программу управления выходом, приведен на рис. 7.

Блок программы управления выходом

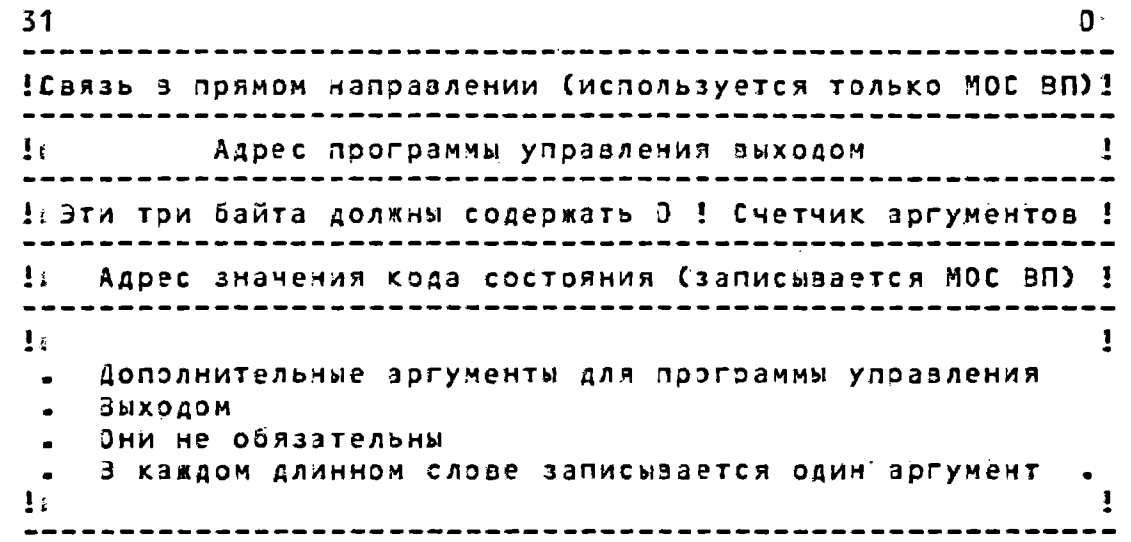


Рис. 7

FAB (Блок доступа к файлу) -

структура, обозначающая блок доступа к файлу СУД-32 (документ [1]);

FILE_PROTECTION (защита файла) -

слово без знака, представляющее из себя 16-битовую маску, которая определяет защиту файла. Эта маска содержит четыре 4-битовых поля, каждое из которых задает уровни защиты, для доступа к файлу одной из четырех групп пользователей. Справа налево поля задают доступ (1) системных пользователей, (2) владельца файла (3) пользователей, относящихся к той же группе кодов идентификации пользователей (UIC), что и владелец файла, и (4) все остальные пользователи. Каждое поле задает, считая биты справа налево, (1) доступ на чтение, (2) доступ на запись, (3) доступ на исполнение

, (4). доступ на удаление. Биты, содержащие "1", указывают, что доступ запрещен. На рис. 8. представлена 16-битовая маска защиты файла.

Маска защиты файла

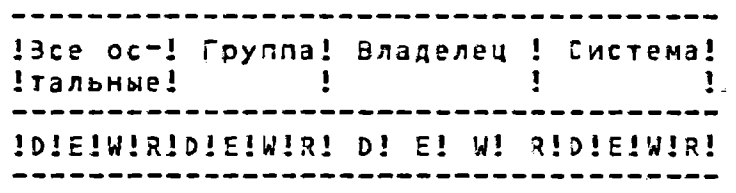


Рис. 8

FLOATING_POINT (плавающая запятая) -

один из станадартных типов данных операционной системы МОС ЭП с плавающей запятой. Таких типов может быть Четыре:

1) число с плавающей запятой F-формата;

2) число с плавающей запятой G-формата;

3) число с плавзющей запятой D-формата;

4) число с плавающей запятой H-формата.

Формат числа с плавающей запятой F-формата представлен на рис. 9.

Формат числа с плавающей запятой F-формата

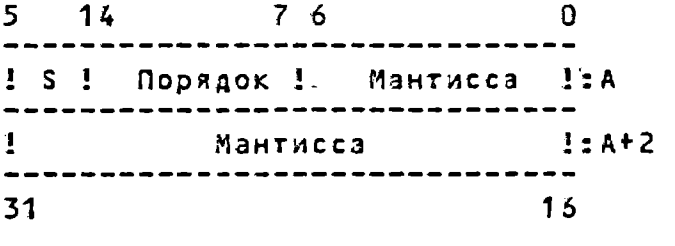


Рис. 9

Формат числа с плавающей запятой D-формата приведен на рис. 10.

Формат числа с плавающей D-формата

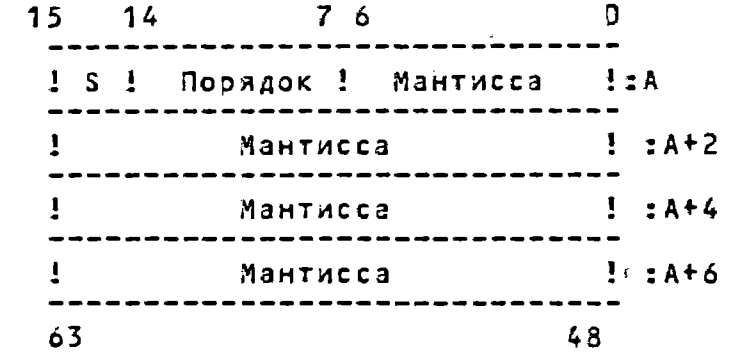


Рис. 10

Формат числа с плавающей запятой G_формата приведен на рис. 11.

Формат числа с плавающей запятой G-формата

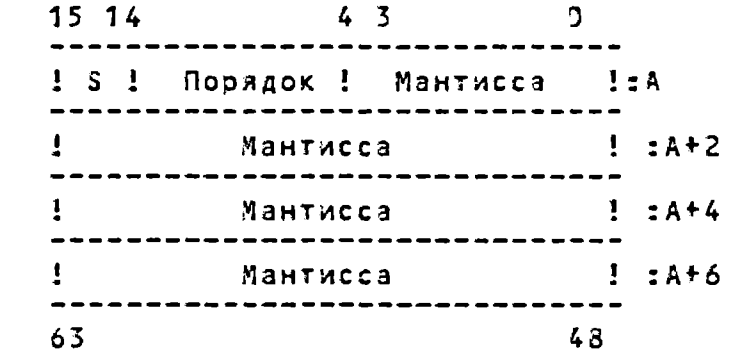


Рис. 11

Формат числа с плавающей запятой H_формата приведен на рис. 12.

Формат числа с плавающей запятой H-формата

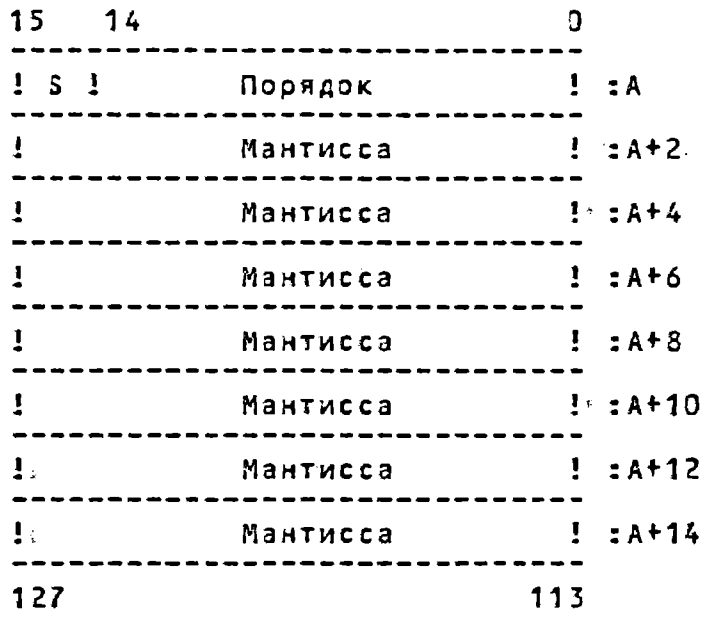


Рис. 12

FUNCTION_CODE (код функции) -

длинное слово, задающее операции, которые должна выполнить процедура. Это длинное слово имеет два поля длиной в слово: первое поле является числом, задающим основную операцию; второе поле представляет из себя маску или вектор битов, задающий различные подоперации внутри основной операции;

IO_STATUS_BLOCK (блок состояния ввода-вывода) -

структура размером в квадрослово, содержащая информацию, возвращаемую программой, которая завершается асинхронно. Возвращаемая информация меняется в зависимости от конкретной программы. На рис. 13 показан формат информации, который записывается в блок IOSB (состояния ввода-вывода) для программы SYS$QIO.

Формат блока состояния ввода-вывода

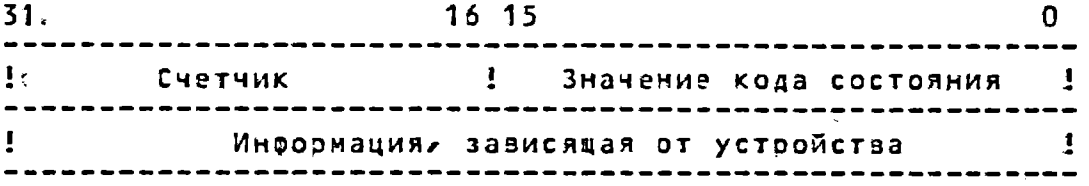


Рис. 13

В данном формате первое слово содержит значение кода состояния, индицирующее успешное или неуспешное завершение операции и используются те же значения кодов состояния, что и для всех других программ системного обслуживания, например, SS$NORMAL указывает на успешное завершение операции. Второе слово содержит число байтов, реально перемещенных в операции ввода-вывода. Для некоторых устройств заполняется только слово счетчика (документ [2]). Второе длинное слово содержит возвращаемую информацию, зависящую от конкретного устройства. Чтобы гарантировать успешное завершение операции ввода-вывода и целостность переданных данных, необходимо проверять блок IOSB после запроса на ввод-вывод, в особенности при выполнении функций ввода-вывода, зависящих от устройств;

ITEM_LIST_2 (список элементов) -

структура, состоящая из одного или более дескрипторов элементов. Последним в списке элементов является длинное слово, содержащее 0. Каждый дескриптор элемента представляет из себя структуру длиной в 2 длинных сло-

ва, которая содержит три поля. На рис. 14 показан один дескриптор элемента:

Дескриптор элемента

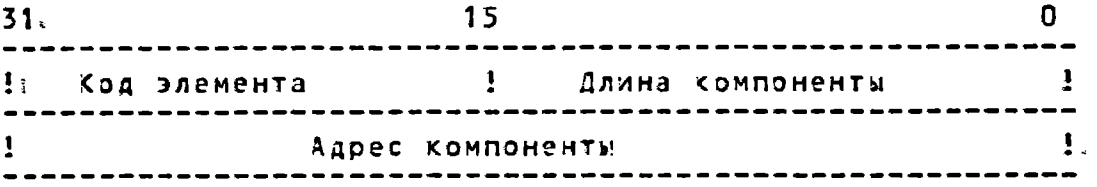


Рис. 14

В первом поле длиной в слово программа системного обслуживания записывает длину (в символах) затребованной компоненты. Если программа системного обслуживания не находит нужной компоненты, то она возвращает в это поле и в поле адреса компоненты значение 0. Второе поле содержит символический код длиной в слово, представляемый пользователем. Данный код задает требуемую компоненту. Коды элементов определяются макрокомандами, специфичными для данной программы системного обслуживания. Третье поле имеет длину равную длинному слову и в нем программа системного обслуживания записывает начальный адрес компоненты. Данный адрес содержится внутри самой входной строки;

ITEM_LIST_3 (список элементов) -

структура, состоящая из одного или более дескрипторов элементов. Последним в этом списке является длинное слово, содержащее 0. Каждый дескриптор элемента представляет из себя структуру из 3-х длинных слов, которая содержит четыре поля. На рис. 15 представлен формат

одного дескриптора элемента.

Дескриптор элемента

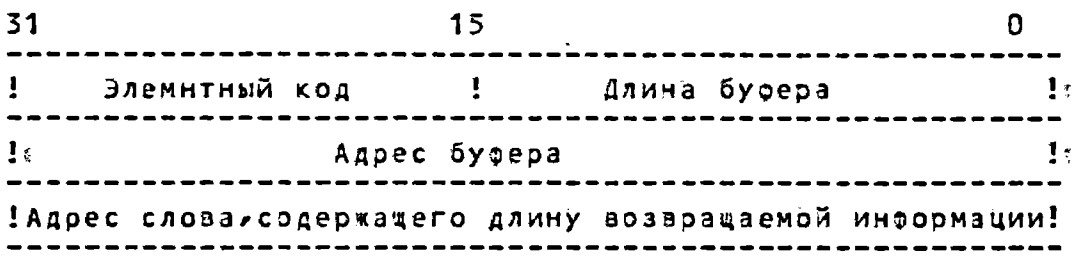


Рис. 15

Первое поле является словом, содержащим представленное пользователем целое число, которое задает длину буфера (в байтах), в который программа системного обслуживания записывает информацию. Требуемая длина буфера зависит от элементного кода, указанного в соответствующем поле дескриптора элемента. Если длина буфера слишком мала, то программа системного обслуживания усекает данные. Второе поле является словом, содержащим предоставляемый пользователем символический код, указывающий тот элемент информации, который должна вернуть программа системного обслуживания. Эти коды определяются в макрокомандах, связанных с данной программой системного обслуживания. Третье поле является длинным словом, содержащим представляемый пользователем адрес буфера, в который программа системного обслуживания записывает информацию. Четвертое поле является длинным словом, содержащим предоставляемый пользователем адрес слова, в которое программа системного обслуживания записывает длину в байтах той инфор-

мации, которую она реально возвращает;

IMET_QUOTA_LIST (список квот) -

структура, содержащая один или более дескрипторов квот. Список заканчивается байтом, содержащим величину, заданную символическим именем PQL_LISTEND. Каждый дескриптор квот состоит из 1-байтового имени квоты, за которым следует длинное слово без знака, содержащее значение для этой квоты;

LOCK_ID (идентификатор захвата) -

длинное слово, содержащее целое число без знака, определяющее идентификатор захвата. Данный идентификатор назначается захвату средством управления захватом, когда разрешается захват;

LOCK_STATUS_BLOCK (блок состояния захвата) -

структура, в которую средство управления захватом записывает информацию состояния захвата. Блок состояния захвата всегда содержит по крайней мере два длинных слова. Первое слово из первого длинного слова Содержит значение кода состояния. Второе слово из первого длинного слова неиспользуется, а второе длинное слово содержит идентификатор захвата. Блок состояния захвата получает конечное значение кода состояния и идентификацию захвата, а также может содержать блок значений захвата. Когда запрос ставится в очередь, идентификация захвата запоминается в блоке состояния захвата, даже если захват не был разрешен. Это позволяет программе выводить из очереди неразрешенные зах-

заты. Значение кода состояния помещается в блок состояния захвата, только когда захват разрещается (или если при разрешении захвата возникает ошибка). На рис. 16 приведен блок состояния захвата, который включает необязательный 16-байтовый блок значений захвата:

Блок состояния захвата

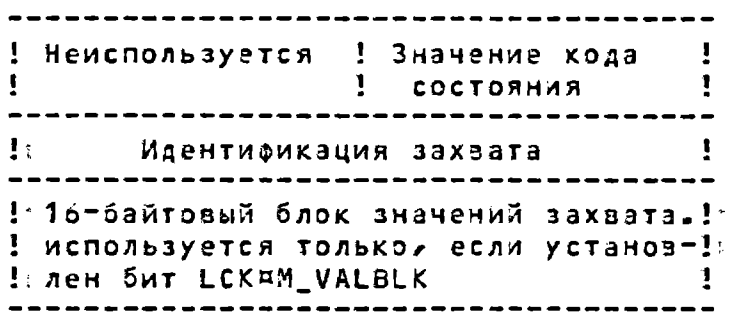


Рис. 16

LOCK_VALUE_BLOCK (блок значений захвата) -

16-байтовый блок, который по запросу средства управления захватом может быть включен в блок состояния захвата. Содержимое блока значений захвата определяется пользователем и не интерпретируется средством управления захвата

LOGICAL_NAME (логическое имя) -

строка, содержащая от 1 до 255 символов, которая идентифицирует логическое имя или эквивалентное имя, которым должна манипулировать программа обслуживания логических имен операционной системы МОС ВП. Логические имена, которые обозначают специфические об'екты операционной системы МОС ВП, имеют свои собственные типы, например, логическое имя, идентифицирующее устройство,

имеет тип "DEVICE_NAME" (имя устройства);

LONGWORD_SIGNED (длинное слово со знаком) -

данный тип данных МОС ЭП совпадает с типом данных "целое число (со знаком) размером в длинное слово", представленном в табл. 5;

LONGWORD_UNSIGNED (длинное слово без знака) -

данный тип данных МОС ЭП совпадает с типом данных "длинное слово (без знака)", представленном в табл. 5;

MASK_BYTE (маска размером в байт) -

байт без знака, в котором каждый бит интерпретируется вызываемой программой. Маску также называют набором "флагов" или "маской битов";

MASK_LONGWORD (маска размером с длинное слово) -

длинное слово без знака, в котором каждый бит интерпретируется вызываемой программой. Маску также называют набором "флагов" или "маской битов";

MASK_QUADWORD (маска размером в квадрослово) -

квадрослово без знака, в котором каждый бит интерпретируется вызываемой программой. Маска также называется набором "флагов" или "маской битов";

MASK_WORD (маска размером в слово) -

слово без знака, в котором каждый бит интерпретируется вызываемой программой. Маску также называют набором "флагов" или "маской битов";

NULL_ARG (нулевой аргумент) -

длинное слово без знака, обозначающее "нулевой аргумент". "нулевой аргумент" - это аргумент, единственное

назначение которого - занимать место в списке аргументов;

OCTAWORD_SIGNED (октаслово со знаком) -

данный тип данных МОС ВП совпадает с типом данных "целое число (со знаком) размером в октаслово", представленном в табл. 5;

OCTAWORD_UNSIGNED (октаслово без знака) -

данный тип данных МОС ВП совпадает с типом данных "октаслово (без знака)", представленном в табл. 5;

PAGE_PROTECTION (защиты страницы) -

длинное слово без знака, определяющее защиту страниц, которая используется аппаратными средствами операционной системы МОС ВП. Значения, определяющие защиту, указываются битами с 0 по 3; биты с 4 по 31 игнорируются. Макрокоманда $PRTDEF определяет символические имена для кодов защиты. Список имен представлен в табл. 4.

Таблица 4

Имя	Описание
PRT$C_NA	Нет доступа
PRT$C_KR	Только чтение в режиме ядра
PRT$C_KW	Запись в режиме ядра
PRT$C_ER	Только чтение в режиме управления
PRT$C_EW	Запись в режиме управления

Продолжение табл. 4

Имя	Описание
PRT#C_SR	Только чтение в режиме супервизора
PRT#C_SW	Запись в режиме супервизора
PRT#C_UR	Только чтение в режиме пользователя
PRT#C_UW	Запись в режиме пользователя
PRT#C_ERKW	Чтение в режиме управления, запись в режиче ядра
PRT#C_SRKW	Чтение в режиме супервизора, запись в режиме ядра
PRT#C_SREW	Чтение в режиме супервизора, запись в режиме управления
PRT#C_URKW	Чтение в режиме пользователя, запись в режиме ядра
PRT#C_UREW	Чтение в режиме пользователя, запись в режиме управления
PRT#C_URSW	Чтение в режиме пользователя, запись в режиме супервизора

Если значение, определяющее защиту, задано равным 0, то по умолчанию разрешено только чтение в режиме ядра;

PROCEDURE (процедура) -

длинное слово без знака, определяющее маску входа в процедуру, которая не должна вызываться на уровне прерывания AST (аргументы, задающие процедуры, которые должны вызываться на уровне AST, имеют тип данных

"AST_PROCEDURe";

PROCESS_ID (идентификатор процесса) -

целое число без знака размером в длинное слово, определяющее идентификатор процесса (PID). Этот идентификатор назначается процессу операционной системой МОС ВП при создании процесса;

PROCESS_NAME (имя процесса) -

строка, содержащая от 1 до 15 символов, которая определяет имя процесса;

QJADRO_SIGNED (квадрослово со знаком) -

этот тип данных МОС ВП совпадает с типом данных "целое число (без знака) размером в квадрослово", представленном в табл. 5;

QJADRO_UNSIGNED (квадрослово без знака) -

этот тип данных МОС ВП совпадает с типом данных "квадрослово без знака", представленном в табл. 5;

RIGHTS_HOLDER (держатель прав) -

квадрослово без знака, задающее права пользователя на доступ к системным об'ектам. Это квадрослово состоит из двух полей: первое - длинное слово без знака, содержащее идентификатор (тип данных МОС ВП RIGHTS_ID), а второе - битовая маска размером в длинное слово, в которой каждый бит указывает право доступа. На рис. 17 приведен формат держателя прав.

Формат держателя прав

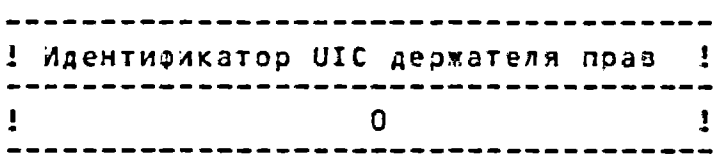


Рис. 17

RIGHTS_ID (идентификатор прав) -

длинное слово без знака, означающее идентификатор прав. Идентификаторы имеют два формата в базе данных прав:

1) формат UIC (тип данных МОС ВП "UIC");

2) формат ID.

старшие биты значения идентификатора указывают формат идентификатора. Если два старших бита имеют значение 0, то это идентификатор формата UIC, если бит 31 установлен, то это идентификатор формата ID. Биты с 28 по 30 не используются. Остальные биты указывают значение идентификатора. На рис. 18 представлен формат ID идентификатора прав.

Рис. 18

Для операционной системы МОС ВП идентификатор является двоичным числом. Чтобы запросить использование идентификатора, операционная система МОС ВП транслирует

двоичный идентификатор в имя идентификатора. Двоичное значение и имя идентификатора связаны в базе данных прав. Имя идентификатора состоит из алфавитно-цифровых символов, из которых по крайней мере один должен быть не цифровым. Число символов может меняться от 1 до 31. Имя идентификатора не может состоять целиком из цифровых символов. Оно может включать символы от A до Z. Знаки денежной единицы (¤) и подчеркивания (_), а также цифры от 0 до 9. Любая строчная буква автоматически преобразовывается в прописную;

RAB (блок доступа к записи) -

структура, содержащая блок доступа к записи системы управления данными СУД-32 (см. Документ [1]);

SECTION_ID (идентификатор секции) -

квадрослово без знака, означающее идентификатор глобальной секции. В данном идентификаторе указывается версия глобальной секции и критерий, который должен использоваться при поиске данной секции;

SECTION_NAME (имя секции) -

строка, содержащая от 1 до 43 символов, которая обозначает имя глобальной секции. Данная строка символов может быть логическим именем, но при этом она должна транслироваться в допустимое имя глобальной секции (раздел 11);

SYSTEM_ACCESS_ID (идентификатор доступа к системе) -

квадрослово без знака, определяющее значение идентификации системы, которое должно быть связано с базой

данных прав;

TIME_NAME (время) -

строка символов, содержащая значение времени в формате МОС ВП;

UIC (код идентификации пользователя) -

длинное слово без знака, содержащее код идентификации пользователя (UIC). Каждый код UIC уникален и представляет пользователя операционной системы МОС ВП. Идентификатор UIC содержит два старших бита, которые обозначают формат, поле члена и поле группы. Диапазон номеров членов составляет от 0 до 65534, диапазон номеров групп - от 1 до 16832. На рис. 19 показан формат кода UIC.

Формат кода UIC

Рис. 19

USER_ARG (пользовательский аргумент) -

длинное слово без знака, содержащее аргумент, определенный пользователем. Данное длинное слово передается программе в качестве аргумента, но содержимое этого длинного слова определяется и интерпретируется пользователем;

VARYING_ARG (переменный аргумент) -

длинное слово без знака, содержащее переменный аргумент. Данный переменный аргумент может иметь различные

типы в зависимости от спецификаций, сделанных для других аргументов при вызове;

VECTOR_BYTE_SIGNED (вектор из байтов со знаком) - однородный массив, всеми элементами которого являются байты со знаком;

VECTOR_BYTE_UNSIGNED (вектор из байтов без знака) - однородный массив, всеми элементами которого являются байты без знака;

VECTOR_LONGWORD_SIGNED (вектор из длинных слов со знаком) - однородный массив, всеми элементами которого являются длинные слова со знаком;

VECTOR_LONGWORD_UNSIGNED (вектор из длинных слов без знака) однородный массив, всеми элементами которого являются длинные слова без знаков;

VECTOR_QUADWORD_SIGNED (вектор из квадрослов со знаком) - однородный массив, всеми элементами которого являются квадрослова со знаком;

VECTOR_QUADWORD_UNSIGNED (вектор из квадрослов без знака) - однородный массив, всеми элементами которого являются квадрослова без знака;

VECTOR_WORD_SIGNED (вектор из слов со знаком) - однородный массив, всеми элементами которого являются слова со знаком;

VECTOR_WORD_UNSIGNED (вектор из слов без знака) - однородный массив, всеми элементами которого являются слова без знака;

WORD_SIGNED (слово со знаком) -

тип данных МОС ВП "слово со знаком" совпадает с типом данных "целое число (со знаком) длиной в слово", представленном в табл. 5;

WORD_UNSIGNED (слово без знака) -

тип данных МОС ВП "слово без знака" совпадает с типом данных "целое число (без знака) длиной в слово", представленном в табл. 5.

1.1.2.2. Элемент "тип"

Когда вызывающая программа передает аргументы программе системного обслуживания, последняя ожидает, чтобы аргументы были данными конкретного типа. В описании программ системного обслуживания указаны ожидаемые типы данных для каждого аргумента.

Аргумент не имеет типа данных, скорее данные, указанные аргументом, имеют тип данных. Аргумент является средством для передачи данных вызываемой программе.

Вызовы программ приводят к построению списка аргументов. Список аргументов представляет из себя вектор из длинных слов. Первое длинное слово содержит счетчик числа остальных длинных слов, а каждое из этих остальных длинных слов является одним из аргументов. Таким образом, аргумент - это одно длинное слово в списке аргументов.

Фраза "тип данного аргумента" часто используется для описания типа того данного, которое указано аргументом. Эта терминалогия используется, поскольку она проще и понятнее,

чем вполне точная фраза "тип того данного, которое указано аргументом".

Табл. 5 содержит типы данных, допустимые для вызовов процедур операционной системы МОС ВП.

Таблица 5

Стандартные типы данных операционной системы МОС ВП

Тип данного	Символический код
Абсолютные дата и время	DSC¤K_DTYPE_ADT
Целое число (со знаком) размером в байт	DSC¤K_DTYPE_B
Максимальная величина метки	DSC¤K_DTYPE_BLV
Максимальная величина процедуры	DSC¤K_DTYPE_BPV
Байт (без знака)	DSC¤K_DTYPE_BU
Число с плавающей запятой D-формата	DSC¤K_DTYPE_D
Комплексное число с плавающей запятой D-формата	DSC¤K_DTYPE_DC
Дескриптор	DSC¤K_DTYPE_DSC
Число с плавающей запятой F-формата	DSC¤K_DTYPE_F
Комплексное число с плавающей запятой F-формата	DSC¤K_DTYPE_FC
Число с плавающей запятой G-формата	DSC¤K_DTYPE_G
Комплексное число с плавающей запятой G-формата	DSC¤K_DTYPE_GC
Число с плавающей запятой H-формата	DSC¤K_DTYPE_H
Комплексное число с плавающей запятой H-формата	DSC¤K_DTYPE_HC

Продолжение табл. 5

Тип данного	Символический код
Целое число (со знаком) размером в длинное слово	DSC$K_DTYPE_L
Длинное слово без знака	DSC$K_DTYPE_LU
Числовая строка, знак слева отдельно	DSC$K_DTYPE_NL
Числовая строка, знак слева встроенный	DSC$K_DTYPE_NLO
Числовая строка, знак справа отдельно	DSC$K_DTYPE_NR
Числовая строка, знак справа встроенный	DSC$K_DTYPE_NRO
Числовая строка без знака	DSC$K_DTYPE_NU
Числовая строка, зонированный знак	DSC$K_DTYPE_NZ
Целое число (со знаком) размером в октаслово	DSC$K_DTYPE_O
Октаслово без знака	DSC$K_DTYPE_OU
Строка упакованного десятичного числа	DSC$K_DTYPE_P
Целое число (со знаком), размером в квадрослово	DSC$K_DTYPE_Q
Квадрослово без знака	DSC$K_DTYPE_QU
Строка символов	DSC$K_DTYPE_T
Выравненная строка битов	DSC$K_DTYPE_V
Переменная строка символов	DSC$K_DTYPE_VT
Невыравненная строка битов	DSC$K_DTYPE_VU
Целое число (со знаком) размером в слово	DSC$K_DTYPE_W
Слово без знака	DSC$K_DTYPE_WU
Неспецифицирован	DSC$K_DTYPE_Z

Продолжение табл. 5

Тип данного	Символический код
Маска входа в процедуру	DSC$K_DTYPE_ZEM
Последовательность инструкций	DSC$K_DTYPE_ZI

1.1.2.3. Элемент "доступ"

Элемент "доступ" описывает способ, которым вызванная программа получает доступ к данным, задаваемым аргументом. В основном используются следующие три метода доступа:

1) только чтение. Данные, с которыми работает программа, или данные, которые требуются программе для выполнения операций, должны считываться вызываемой программой. Такие данные также называются входными данными. Когда аргумент задает входные данные, то элемент "доступ" показывает "только чтение".

Термин "только" необходим, чтобы показать, что программа не выполняет обе операции - чтение и запись (т.е. Модификацию) входных данных. Таким образом, входные данные, представленные переменной, сохраняются, когда вызванная программа завершает выполнение;

2) только запись. Данные, которые вызываемая программа возвращает вызывающей программе, должны быть записаны в то место памяти, откуда вызывающая программа может их достать. Такие данные также называются выходными. Если аргумент задает выходные данные, то элемент "доступ" содержит "толь-

ко запись".

Термин "только" необходим, чтобы показать, что вызываемая программа не читает содержимое памяти ни до, ни после того, как она записывает туда данные;

3) модификация. Если аргумент задает данные, которые и читаются и записываются вызываемой программой, то элемент "доступ" содержит "модификация". В этом случае вызываемая программа читает входные данные, которые она использует в своих операциях, а затем на место входных данных записывает результат (т.е. выходные данные) операции. Таким образом, после завершения выполнения вызываемой программы входные данные, заданные аргументом, теряются.

Полный список типов доступа, допустимых стандартами вызова процедур операционной системы МОС ВП:

1) только чтение;

2) только запись;

3) модификация;

4) вызов функции (перед возвратом);

5) выполнение инструкции JMP после развертывания;

6) вызов после развертывания стека;

7) вызов без развертывания стека.

1.1.2.4. Элемент "механизм"

Способ, которым аргумент задает реальные данные, используемые вызываемой программой, определяется в терминах механизма передачи аргумента. Имеется три механизма передачи:

1) по значению. Если аргумент размером в длинное слово из списка аргументов содержит реальные данные, которые должны использоваться программой, то реальные данные передаются программе "по значению". В этом случае аргумент размером в длинное слово содержит реальные данные, аргумент и есть реальные данные. Поскольку аргумент имеет размер в длинное слово, то по значению могут передаваться только те данные, которые могут быть представлены длинным словом;

2) по ссылке. Если аргумент размером в длинное слово из списка аргументов содержит адрес данных, которые должны использоваться программой, то данные передаются "по ссылке". В этом случае аргумент является указателем на данные;

3) по дескриптору. Если аргумент размером в длинное слово из списка аргументов содержит адрес дескриптора, то данные передаются "по дескриптору". Дескриптор состоит из двух или более длинных слов (в зависимости от типа используемого дескриптора), в которых описывается тип данных, которые должны использоваться вызываемой программой. В этом случае аргумент является указателем на дескриптор, который в свою очередь является указателем на реальные данные.

Список механизмов передачи, допустимые стандартом вызова процедур операционной системы МОС ВП, приведен в табл. 6.

Таблица 6

Механизм передачи	Код дескриптора
По значению	
По ссылке	
По дескриптору	
По ссылке, массив ссылок	
По дескриптору, фиксированная длина	DSC$K_CLASS_S
По дескриптору, строка переменной длины	DSC$K_CLASS_D
По дескриптору, массив	DSC$K_CLASS_A
По дескриптору, программа	DSC$K_CLASS_P
По дескриптору, десятичная строка	DSC$K_CLASS_SD
По дескриптору, несплошной массив	DSC$K_CLASS_NCA
По дескриптору, переменная строка	DSC$K_CLASS_VS
По дескриптору, массив переменных строк	DSC$K_CLASS_VSA
По дескриптору, невыровненная строка битов	DSC$K_CLASS_UBS
По дескриптору, невыровненный массив битов	DSC$K_CLASS_UBA
По дескриптору, строка с границами	DSC$K_CLASS_SB
По дескриптору, невыровненная строка битов с границами	DSC$K_CLASS_UBSB

1.1.2.5. Элемент "пояснительный текст"

Для каждого элемента "тип", "доступ" и "механизм" следует пояснительный текст, который содержит следующую информацию:

1) начальное предложение, которое описывает:

- физическую сущность данных, указанных аргументом;

- способ, которым программа использует эти данные.

Например, если аргумент указывает число, которое программа должна преобразовать в другой тип данных, то начальное предложение могло бы иметь вид: "число, которое должно преобразовываться в такой-то тип данных";

2) предложение, выражающее отношение между аргументом и данными, которые он специфицирует. Эти отношения являются используемым механизмом передачи данных.

Если испольуется механизм передачи "по значению", то это предложение записывается следующим образом: "аргумент XXX содержит такие-то данные".

Если используется механизм передачи "по ссылке", то это предложение записывается вроде следующего: "аргумент XXX является адресом таких-то данных".

Если используется механизм передачи "по дескриптору", то это предложение записывается вроде следующего: "аргумент XXX является адресом дескриптора, указывающего на такие-то данные".

1.1.3. Заголовок "возвращаемые значения кодов состояния".

Значение кода состояния представляет из себя длинное слово без знака и имеет следующие различные применения в архитектуре операционной системы МОС ВП:

1) индицирует успешное или неуспешное завершение вызванной программы;

2) описывает исключительные состояния;

3) идентифицирует системные сообщения;

4) сообщает об успешном или неуспешном выполнении программы на уровне диалогового командного языка.

Под заголовком "возвращаемые значения кодов состояний" в списке из двух колонок дан символический код для каждого значения кода состояния, которое может вернуть программа, и соответствующее описание. Описание поясняет, означает ли данное значение кода состояния успех или неудачу, и, если это неудача, то какие действия пользователя могли вызвать неудачу, а также, что можно сделать, чтобы исправить ситуацию.

Символические коды для значений кодов состояния определены операционной системой МОС ВП. Символический код, определенный для каждого значения кода состояния, соответствует числу, то есть идентичен значению кода состояния размером в длинное слово, которое интерпретируется как число. Хотя значение кода состояния состоит из различных полей, каждое из которых может интерпретироваться отдельно как конкретная информация, все значение кода состояния раз-

мером в длинное слово как целое может интерпретироваться как целое число без знака размером в длинное слово, и данное длинное слово имеет эквивалентный символический код.

Если вызываемая программа генерирует исключительное состояние, то данное исключительное состояние включено. Исключительное состояние затем обрабатывается программой обработки исключительных состояний (пользовательской или системной). В зависимости от конкретного исключительного состояния и от программы обработки исключительных состояний, вызываемая программа либо продолжает нормальное выполнение, либо завершается по ошибке.

Под заголовком "возвращаемые значения кодов состояния" описываются значения кодов состояния, возвращаемые программе, когда она заканчивает выполнение без генерации исключительного состояния.

1.1.4. Заголовок "значения кодов состояния, возвращаемые в блок состояния ввода-вывода"

Если вызываемая программа возвращает значение кода состояния в блок состояния ввода-вывода, то возможные значения этих кодов перечисляются под заголовком "значения кодов состояния, возвращаемые в блок состояния ввода-вывода".

Некоторые программы системного обслуживания завершаются асинхронно, то есть они возвращают управление в точку вызова сразу после того, как вызов программы системного

обслуживания успешно ставится в очередь, но перед завершением операции, которая должна быть выполнена программой системного обслуживания. Это позволяет вызывающей программе продолжать выполнение, в то время, как программа системного обслуживания сама еще выполняется. Все программы системного обслуживания, которые завершаются асинхронно, имеют аргументы, указывающие блок состояния ввода-вывода. После того, как операция программы системного обслуживания завершается, значение кода состояния, указывающее состояние завершения операции, записывается в блок состояния ввода-вывода.

Первое слово в блоке состояния ввода-вывода получает значение кода состояния для конечного состояния завершения асинхронной программы системного обслуживания. Представление значения кода размером в длинное слово в поле размером в слово возможно для программы системного обслуживания, поскольку старшее слово значения кода состояния для программы системного обслуживания содержит нуль.

Одно поле в значении кода состояния указывает, какое средство сгенерировало код состояния. Данное поле расположено в старшем слове длинного слова, содержащего значение кода состояния. Этот факт дает возможность представить значение кода состояния, сгенерированного системным средством (включая все программы системного обслуживания), в одном слове, а не в длинном слове, так как все биты старшего слова имеют значение нуль.

2. Вызов программ системного обслуживания

При вызове программ системного обслуживания используются стандартные соглашения вызова процедур операционной системы МОС ВП. Языки программирования, которые генерируют инструкции основного режима операционной системы МОС ВП, предоставляют механизм для указания вызова программ.

При кодировании вызова программы системного обслуживания необходимо задать все аргументы, которые требует данная программа системного обслуживания.

После завершения выполнения программы системного обслуживания, она возвращает управление вызывающей программе вместе с кодом состояния. В месте вызова следует проанализировать значение кода состояния, чтобы определить, успешно или неуспешно прошел вызов программы системного обслуживания. При этом программа, если необходимо, может изменить последовательность выполнения.

Для программиста, программирующего на языке макроассемблер, необходимо прочитать подраздел 2.4, где подробно описано, как писать инструкции, которые генерируют вызовы программ системного обслуживания.

Как программисту, программирующему на языке макроассемблер, так и программисту на языке программирования высокого уровня, следует прочитать подразделы 2.2, 2.6. В подразделе 2.2 представлена информация по спецификации аргументов для программ системного облсуживания. В подразделе 2.6 описаны методы проверки состояния возврата из программы

системного обслуживания.

Программисту, программирующему на языке программирования высокого уровня следует прочитать п. 1.2.7, где приведена информация о том, как вызывать программы системного обслуживания из языков программирования высокого уровня. Детальная информация и примеры приведены в руководствах пользователей по соответствующему языку.

Макрокоманды программ системного обслуживания генерируют список аргументов и инструкции CALL для вызова программ системного обслуживания. Данные макрокоманды находятся в системной библиотеке SYS$LIBRARY:STARLET.MLB. Библиотека просматривается автоматически для неразрешенных ссылок при ассемблировании исходной программы.

Для понимания материала, представленного в этом разделе, требуется знание правил макроязыка операционной системы МОС ВП при программировании на языке макроассемблера (документы [3] и [4]).

2.1. Программы системного обслуживания и целостность системы

Многие программы системного обслуживания доступны и удобны для прикладных программ, однако использование некоторых из этих программ должно быть ограничено, чтобы защитить работоспособность операционной системы МОС ВП и целостность процессов пользователей.

Например, поскольку создание постоянных почтовых ящиков использует динамическую память системы, то неограничен-

ное пользование постоянными почтовыми ящиками может уменьшить количество памяти, доступной другим пользователям. Поэтому и контролируется возможность создания постоянных почтовых ящиков. Пользователю специально назначают привилегию использовать программу системного обслуживания "создать почтовый ящик" (=CREMBX) для создания постоянного почтового ящика.

2.1.1. Привилегии пользователей

Администратор системы, который сопровождает файл авторизации пользователей для операционной системы МОС ВП, предоставляет привилегии для пользования защищенными программами системного обслуживания. Файл авторизации пользователей содержит кроме основной информации о пользователе список специфических привилегий и квоты ресурсов.

Когда пользователь подключается к операционной системе МОС ВП, то назначенные ему привилегии и квоты связываются с процессом, который создается для него. Данные привилегии и квоты относятся к каждому образу, который выполняет процесс.

Когда образ делает вызов программы системного обслуживания, которая защищена привилегиями, проверяется список привилегий. Если пользователю назначена требуемая специфическая привилегия, то образу разрешается выполнить программу системного обслуживания, в противном случае возвращается код состояния, указывающий на ошибку.

2.1.2. Квоты ресурсов

Многие программы системного обслуживания требуют для своего выполнения определенных служебных ресурсов. Служебные ресурсы включают системную динамическую память и квоты процесса для операций ввода-вывода. Когда вызывается программа системного обслуживания, которая использует ресурс, контролируемый квотой, то проверяется квота процесса на требуемый ресурс. Если процесс превысил свою квоту, или процессу не была выделена квота, то может быть возвращен код состояния, указывающий на ошибку. Обычно, когда вызывается программа системного обслуживания, а требуемый ресурс недоступен, то процесс помещается в состояние ожидания до тех пор, пока ресурс не станет доступным. Затем программа системного обслуживания завершает выполнение. Данный режим называется режимом ожидания ресурса.

Однако в среде реального времени состояние ожидания программы может быть нежелательным. В таких случаях программист может решить отменить режим ожидания ресурса, так что если требуемый ресурс недоступен, то управление сразу возвращается в вызывающую программу с кодом состояния, указывающим на ошибку. Режим ожидания ресурса можно отключить (или включить) с помощью программы системного обслуживания "установить режим ожидания ресурса" ($SETRWM).

То, как программа реагирует на недоступность ресурса, зависит в сильной степени от конкретной программы пользователя и от вызываемой программы системного обслуживания. В некоторых случаях программа может продолжить выполнение и

попытаться заново вызвать программу системного обслуживания позднее. В других случаях может понадобиться только отметить, что программа должна была перейти в состояние ожидания.

2.1.3. Режим доступа

Процесс может выполняться в одном из четырех режимов доступа:

1) режим пользователя;

2) режим супервизора;

3) режим управления;

4) режим ядра.

Режим доступа определяет возможность процесса получить доступ к страницам виртуальной памяти. Каждая страница имеет связанный с ней код защиты, задающий тип доступа - читать, писать или нет доступа - допустимый для каждого режима (документы [5] и [6]).

В большинстве случаев написанные пользователем программы выполняются в режиме пользователя. Системные программы, выполняющиеся по запросу пользователя (например, программы системного обслуживания), могут работать в одном из трех остальных режимов доступа, которые являются более привилегированными.

При некоторых вызовах программ системного обслуживания проверяется режим доступа вызывающей программы. Например, если процесс пытается отменить запросы таймера, то он может отменить только те запросы, которые были сделаны из того же

самого или менее привилегированных режимов доступа. Если, скажем, процесс выполняется в режиме пользователя, то он не может отменить запросы таймера, сделанные из режима супервизора, режим управления или ядра, поскольку это более привилегированные режимы доступа.

Многие программы системного обслуживания используют режим доступа, чтобы защитить ресурсы операционной системы МОС ЭП и для этого применяют специальное соглашение по интерпретации аргументов режима доступа. Можно задавать режим доступа, используя числовые значения или символические имена. В табл. 7 перечислены режимы доступа и приведены соответствующие им числовые значения и символические имена.

Таблица 7

Режим доступа	Числовое значение	Символическое имя	Ранг привилегий
Режим ядра	0	PSL$C_KERNEL	Высокий
Режим управления	1	PSL$C_EXEC	
Режим супервизора	2	PSL$C_SUPER	
Режим пользователя	3	PSL$C_USER	Низкий

Программы системного обслуживания, которые допускают аргументы режима доступа, могут только разрешить вызывающим программам задать режим доступа менее привилегированный или

равный по привилегии тому режиму доступа, из которого была вызвана программа системного обслуживания. Если же задается более привилегированный режим доступа, чем у вызывающей программы, то используется менее привилегированный режим доступа.

Чтобы определить, какой режим доступа использовать, операционная система МОС ВП сравнивает указанный режим доступа с тем, из которого программа системного обслуживания была вызвана. Если режимы доступа различны, то всегда используется наименее привилегированный из них. Поскольку результатом этой операции является режим доступа с большим числовым значением (если режим доступа вызывающей программы отличается от указанного режима доступа), то режим доступа "максимизирован".

Поскольку большинство кодов, написанных пользователем, выполняются в режиме пользователя, то можно опускать аргумент режима доступа. По умолчанию данный аргумент имеет значение 0 (режим ядра), а при сравнивании этого значения со значением текущего выполняемого режима доступа (3, режим пользователя) будет использовано наибольшее значение (3).

2.2. Задание аргументов для программ системного обслуживания

Аргументы, которые требуются программе системного обслуживания, приведены в разделе 13 в описании каждой из этих программ. Под заголовком "формат" каждого такого описания указываются позиции и ключевые имена всех аргументов, как

показано в примере 1.

Пример 1.

¤SERVICE ARGa, ARGB, ARGC, ARGD

Формат указывает, что имя программы системного обслуживания есть SERVICE и что она требует задания четырех аргументов, упорядоченных в указанной последовательности, с ключевыми именами ARGa, ARGB, ARGC и ARGD. Для списка аргументов этой программы системного обслуживания необходимо использовать формат, изображенный на рис. 20.

Формат списка аргументов

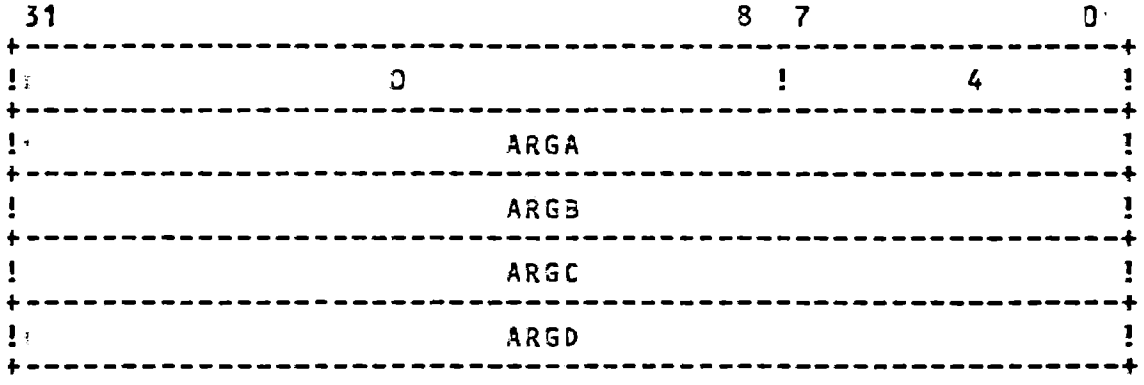


Рис. 20

Все аргументы имеют размер в длинное слово. Первое длинное слово в списке всегда должно содержать в младшем байте количество аргументов в списке. Остальные три байта должны быть равны нулю.

Многие аргументы для вызова программ системного обслуживания необязательны. Они отмечаются в формате макрокоманды квадратными скобками. Например, если второй и третий аргументы макрокоманды ¤SERVICE необязательны, то формат этой макрокоманды имеет вид:

=SERVICE ARGa, [ARGB], [ARGC], ARGD

Если необязательный аргумент опускается в макрокоманде программы системного обслуживания, то макрокоманда использует для такого аргумента значение по умолчанию.

Необязательные аргументы для программы системного обслуживания всегда имеют значение по умолчанию, независимо от того, передаются ли они по значению, по ссылке или по дескриптору. Почти в каждом случае значение по умолчанию необязательного аргумента равно нулю. Макрокоманды, используемые для вызова программ системного обслуживания, позволяют некоторым языкам устанавливать значения по умолчанию, отличные от нуля (например, макроассемблер и язык блисс-32).

Описание необязательного аргумента всегда указывает, какие действия предпринимает программа системного обслуживания, если используется значение по умолчанию.

Аргументы, которые задают адрес возврата, могут быть определены, когда программа системного обслуживания возвращает информацию. Если программа не требует информации, этот необязательный аргумент можно опустить.

2.3. Получение значений для символических кодов

Отдельные программы системного обслуживания имеют символические коды для специальных состояний возврата, смещений в списке аргументов, идентификаторов и флагов, связанных с этими программами. Например, программа системного

обслуживания "создать процесс" (SCREPRC), которая используется для создания подпроцесса или отсоединенного процесса имеет символические коды, связанные с различными привилегиями и квотами, которые могут быть представлены создаваемому процессу.

Макробиблиотека операционной системы МОС ВП, используемая по умолчанию, STARLET.MLB, содержит макроопределения для большинства системных символических имен. При ассемблировании исходной программы, которая вызывает какую-либо из этих макрокоманд, макроассемблер автоматически осуществляет поиск макроопределений в библиотеке STARLET.MLB. Каждое символическое имя имеет числовое значение.

Если язык, используемый пользователем, имеет средство получения значений для этих символических имен, то это об'ясняется в соответствующем руководстве пользователя.

Если же язык не имеет такого средства, то можно сделать следующее:

1) написать короткую программу на макроассемблере, которая содержит требуемую макрокоманду;

2) ассемблировать эту программу и сгенерировать листинг. Используя листинг, найти требуемые символические имена и их шестнадцатеричные значения;

3) определить каждое символическое имя со своим значением в исходной программе.

Например, чтобы использовать программу системного обслуживания "получить информацию о задании/процессе" (SGETJPI) для получения суммарного времени центрального

процессора (в 10-миллисекундных интервалах) заданного процесса, необходимо получить значение, связанное с идентификатором элемента JPI¤_CPUTIM. Это можно сделать следующим способом:

1) написать приведенную программу (пример 1) из трех строк на макроассемблере (данная программа названа JPIDEF.MAR, однако допускается любое другое имя);

Пример 1.

```
.TITLE  JPIDEF   получить значения для ¤JPIDEF
¤JPIDEF GLOBAL
.END
```

2) ассемблировать и скомпоновать программу для создания файла JPIDEF.MAP (пример 2);

Пример 2.

```
¤MACRO  JPIDEF
¤LINK/NOEXE/MAP/FULL    JPIDEF
%LINK-W-USRTFR, IMAGE NL:[].EXE
```

Файл JPIDEF.MAP будет содержать логические имена, определенные макрокомандой ¤JPIDEF;

3) найти значение идентификатора JPI¤_CPUTIM и определить символическое имя в своей программе.

2.4. Вызов программ системного обслуживания из программ на языке макроассемблер

Макрокоманды программ системного обслуживания генерируют список аргументов и инструкции CALL для вызова программ системного обслуживания. Данные макрокоманды находятся

в системной библиотеке SYS¤LIBRARY:STARLET.MLB. Библиотека автоматически просматривается для разрешения ссылок, неразрешенных при ассемблировании исходной программы.

Для понимания материала данного подраздела необходимо знание правил языка программирования макроассемблер.

Каждая программа системного обслуживания имеет четыре связанные с нею макрокоманды. Макрокоманды позволяют пользователю определять символические имена для смещений аргументов, формировать список аргументов для программ системного обслуживания и вызывать программы системного обслуживания. Обобщенные макрокоманды и функции, которые они выполняют, приведены в табл. 8.

Таблица 8

Макрокоманды	Функция
¤имяDEF	Определяет символические имена для смещений в списке аргументов
¤имя	Определяет символические имена для смещений в списке аргументов и формирует список аргументов
¤имя_S	Вызывает программу системного обслуживания и формирует список аргументов
¤имя_G	Вызывает программу системного обслуживания и использует список аргументов, сформированный макрокомандой ¤имя

2.4.1. Использование макрокоманд для формирования списков аргументов

Имеется две обобщенные макрокоманды для формирования списков аргументов программ системного обслуживания: ¤имя и ¤имя_S.

Использование той или иной макрокоманды зависит от того, какую макрокоманду используют для вызова программы системного обслуживания. Если для вызова программы системного обслуживания используется макрокоманда ¤имя_G, то для формирования списка аргументов следует использовать макрокоманду ¤имя. Если для вызова программы системного обслуживания используется макрокоманда ¤имя_S, то эту же макрокоманду можно использовать для формирования списка аргументов.

Если для формирования списка аргументов программы системного обслуживания используются макрокоманды ¤имя_S или ¤имя, то аргументы можно указать одним из трех способов:

1) использовать ключевые слова для описания аргументов. За каждым ключевым словом должен следовать знак равенства (=), а затем значение аргумента;

2) использовать позиционный порядок, где опущенные аргументы отмечаются запятыми в соответствующих позициях. Для необязательных аргументов в конце списка запятую можно опускать;

3) использовать как позиционный порядок, так и ключевые имена (позиционные аргументы должны следовать в списке первыми).

Пример.

```
¤SERVICE ARGA ,[ARGB] ,[ARGC], ARGD
```

Предполагается, что в этом примере аргументы ARGA и ARGB задают числовые значения, а аргументы ARGC и ARGD задают адреса.

В примерах 1 и 2 показаны допустимые способы записи макрокоманды ¤имя_S для вызова программы ¤SERVICE.

Примеры:

1. Использование ключевых слов

```
MYARGD: .LONG   100

        ¤SERVICE_S  ARGB=#0, ARGC=0, ARGA=#1, ARGD=MYARGD
```

2. Задание аргументов в позиционном порядке

```
MYARGD: .LONG   100

        ¤SERVICE_S  #1,,,MYARGD
```

Список аргументов помещается в стек следующим образом:

```
PUSHAL  MYARGD
PUSHL   #0
PUSHL   #0
PUSHL   #1
```

Все аргументы независимо от того, заданы ли они позиционно или с помощью ключевых слов, должны быть допустимыми выражениями языка макроассемблер, поскольку они используются как исходные операнды в инструкциях.

В примерах 3 и 4 показаны правильные способы записи

макрокоманды ¤имя, формирующей список аргументов для последующего вызова программы.

Пример 3.

```
LIST:   ¤SERVICE-
                ARGB=0, -
                ARGC=0, -
                ARGA=1, -
                ARGD=MYARGD
```

Пример 4.

```
LIST:   ¤SERVICE-
                1,,,MYARGD
```

В обоих случаях общий список аргументов имеет вид:

```
LIST:   .LONG   4
        .LONG   1
        .LONG   0
        .LONG   0
        .ADDRESS -
                MYARGD
```

Все аргументы независимо от того, заданы ли они позиционно или с помощью ключевых слов, должны быть такими выражениями, из которых макроассемблер может сгенерировать директивы данных .LONG и .ADDRESS. В отличие от этого аргументы макрокоманды ¤имя_S должны быть правильными выражениями макроассемблера, поскольку они используются как исходные операторы в инструкциях.

2.4.1.1. Соглашения о спецификации аргументов для программ системного обслуживания

Аргументы необходимо специфицировать в соответствии с правилами макроассемблера операционной системы МОС ВП по спецификации и адресации операндов.

Способ спецификации конкретного аргумента зависит от следующих факторов:

1) что требует в качестве аргумента программа системного обслуживания - адрес или значение. Булева величина, число или маска принимаются в качестве аргумента;

2) какая используется макрокоманда программы системного обслуживания.

Если пользователю неизвестно, правильно ли он специфицировал аргумент, который является адресом или значением, он может ассемблировать программу с директивой .LIST MEB, чтобы проверить макрорасширение (см. документ [3]).

2.4.1.2. Определение символических имен для смещений в списке аргументов: ¤имя и ¤имяDEF

К аргументам из списка аргументов можно обращаться по символическому имени. Каждый аргумент в списке аргументов имеет смещение от начала списка. Для численного выражения смещения каждого аргумента определяется символическое имя. Если для обращения к аргументам в списке используются символические имена, то нет необходимости помнить численное

значение смещения (которое основывается на позиции аргумента, показанной в формате макрокоманды).

Имена смещений для списков аргументов всех программ системного обслуживания формируются путем сцепления имени макрокоманды программы системного обслуживания с символами ¤_ и ключевым именем аргумента:

Имя¤_ключевое слово

где имя

- это имя программы системного обслуживания;

ключевое слово

- ключевое имя аргумента.

Подобным образом число аргументов, которое требуется для конкретной макрокоманды, определяется символически следующим образом:

Имя¤_NARGS

Символические имена для смещений в списке аргументов определяются автоматически, как только используется макрокоманда ¤имя для конкретной программы системного обслуживания. Кроме того, можно определить символические имена для списка аргументов программы системного обслуживания, используя макрокоманду ¤имяDEF. Данная макрокоманда не генерирует выполняемый код, она просто определяет символические имена, так что они могут быть использованы позднее в программе.

Пример.

¤QIODEF

Макрокоманда ¤QIODEF определяет символическое имя

QIO¤NARGS и символические имена смещений списка аргументов программы системного обслуживания.

Необходимость в макрокоманде ¤имяDEF может возникнуть, если пользователь специфицирует список аргументов для программы системного обслуживания без использования макрокоманды ¤имя, или если программа обращается к списку аргументов в отдельно ассемблированном модуле.

Например, макрокоманды ¤READEF и ¤READEFDEF определяют следующие значения:

READEF¤_NARGS - число аргументов в списке (2);

READEF¤_EFN - смещение аргумента EFN (4);

READEF¤_STATE - смещение аргумента STATE (8).

Таким образом, для вызова программы системного обслуживания ¤READEF можно специфицировать макрокоманду ¤READEF в целях построения списка аргументов.

Пример 1.

```
READLST: ¤READEF EFN=1,STATE=TEST1
```

Позднее программе может понадобиться использовать другое значение для аргумента STATE при вызове программы системного обслуживания. В примере 2 показано, как это можно выполнить, вызывая макрокоманду ¤имя_G.

Пример 2.

```
    MOVAL       TEST2,READLST+READEF¤_STATE
    ¤READEF_G   READLST
```

Инстукция MOVAL заменяет адрес TEST1 в списке аргументов программы ¤READEF на адрес TEST2. Затем макрокоманда ¤READEF_G вызывает программу системного обслуживания с

модифицированным списком.

2.4.2. Использование макрокоманд для вызова программ системного обслуживания

Имеется две обобщенные макрокоманды для записи вызова программы системного обслуживания:

¤имя_S

¤имя_G

Выбор конкретной макрокоманды зависит от того, как сформирован список аргументов для программы системного обслуживания:

1) макрокоманда ¤имя_S требует, чтобы аргументы для программы системного обслуживания были представлены в макрокоманде этой программы. Макрокоманда генерирует код для занесения списка аргументов в стек вызова при выполнении программы. С помощью этой макрокоманды можно использовать регистры для хранения ссылок на аргументы;

2) макрокоманда ¤имя_G требует, чтобы список аргументов был сформирован в каком-то месте программы и был указан адрес этого списка в качестве аргумента для программы системного обслуживания (имеется макрокоманда для создания списка аргументов для каждой программы системного обслуживания. С помощью этой макрокоманды можно использовать один и тот же список аргументов с необходимыми модификациями для более чем одного вызова макрокоманды).

Макрокоманда ¤имя_S генерирует инструкцию CALLS, а макрокоманда ¤имя_G генерирует инструкцию CALLG. Программы

системного обслуживания вызываются в соответствии со стандартными соглашениями по вызову процедур. Программы системного обслуживания сохраняют все регистры кроме R0 и R1 и восстанавливают сохраненные регистры перед тем, как вернуть управление в точку вызова.

2.4.2.1. Макрокоманда имя_S

Формат

имя_S ARG1,...,ARGN

Макрокоманда генерирует код для помещения аргументов в стек в обратном порядке. Действительные инструкции, используемые для помещения аргументов в стек, определяются следующим образом:

1) если программа системного обслуживания требует в качестве аргумента значение, то генерируется инструкция PUSHL или MOVZWL;

2) если программа системного обслуживания требует адрес в качестве аргумента, то в зависимости от контекста генерируется инструкция PUSHAB, PUSHAW, PUSHAL, PUSHAQ.

Затем макрокоманда генерирует вызов программы системного обслуживания (пример 1).

Пример 1.

```
CALLS   #N, @#SYSимя
```

где #N - число аргументов в стеке.

Поскольку макрокоманда имя_S формирует список аргументов во время выполнения, то адрес и значения можно представлять, используя режимы адресации регистров (пример

2).

Пример 2.

¤READEF_S EFN=#1, STATE=(R10)

Регистр R10 содержит адрес длинного слова, которое принимает информацию о состоянии флагов.

Данная макрокоманда расширяется следующим образом:

```
PUSHAL  (R10)
PUSHAL  #1
CALLS   #2, @#SYS¤READEF
```

2.4.2.2. Макрокоманда ¤имя_G

Формат

¤имя_G метка

где метка - это адрес списка аргументов.

Макрокоманда ¤имя предназначена для создания списка аргументов макрокоманды ¤имя_G (пример 1).

Пример 1.

Метка: ¤имя ARG1,...,ARGN

где метка - символический адрес сгенерированного списка аргументов. Эта метка используется в качестве аргумента в макрокоманде ¤имя_G;

¤имя - это имя макрокоманды программы системного обслуживания;

ARG1,...ARGN - аргументы, которые размещаются в последовательных длинных словах списка аргументов.

Макрокоманда $имя_G (используемая с макрокомандой $имя) особенно полезна для выполнения следующих действий:

1) вызов программ системного обслуживания, имеющих длинные списки аргументов;

2) повторный вызов программ системного обслуживания во время выполнения отдельной программы с тем же самым или в основном с тем же самым списком аргументов.

Формат макрокоманды программы системного обслуживания $READEF показан в примере 2.

Пример 2.

```
$READEF   EFN, STATE
```

Аргумент EFN указывает номер кластера флагов событий, а аргумент STATE указывает адрес длинного слова, которое принимает содержимое кластера.

Данные аргументы можно специфицировать, используя макрокоманду $имя (пример 3).

Пример 3.

```
READLST:
        $READEF EFN=1, -     ;список аргументов для $READEF
                STATE=TESTFLAG
```

Макрокоманда $READEF генерирует код, представленный в примере 4.

Пример 4.

```
READLST:
        .LONG   2           ;список аргументов для $READEF
        .LONG   1
```

.ADDREESS -

TESTFLAG

Теперь для выполнения макрокоманды ¤READEF требуется строка, приведенная в примере 5.

Пример 5.

```
¤READEF_G        READLST
```

Данная макрокоманда генерирует для вызова программы системного обслуживания код, приведенный в примере 6.

Пример 6.

```
CALLG   READLST,0#SYS¤READEF
```

Где SYS¤READEF - имя вектора для точки входа программы системного обслуживания "читать флаги событий". Компоновщик автоматически разрешает адреса точек входа для всех программ системного обслуживания.

2.5. Завершение программ системного обслуживания

После завершения программы системного обслуживания управление возвращается в вызывающую программу. Пользователь может решить, когда и как управление возвращается в его программу, выбирая синхронную или асинхронную формы программ системного обслуживания и включая различные режимы выполнения процесса.

2.5.1. Синхронные и асинхронные программы системного обслуживания

Целый ряд программ системного обслуживания может выполняться либо синхронно, либо асинхронно (например, SYS$GETJPI и SYS$GETJPIW). Символ "W" в конце имени программы системного обслуживания указывает на синхронную версию программы системного обслуживания.

Асинхронная версия программы системного обслуживания ставит запрос в очередь и возвращает управление в вызывающую программу. Вызывающая программа может выполнять операции, пока выполняется программа системного обслуживания. Не проверив, что программа системного обслуживания закончилась, не имеет смысла обращаться к информации, возвращаемой этой программой.

Обычно программа передает асинхронной программе системного обслуживания флаг события и блок управления ввода-вывода. Когда программа системного обслуживания заканчивается, она устанавливает флаг события и помещает в блок ввода-вывода конечное состояние запроса. Чтобы удостовериться, что данная программа системного обслуживания закончилась, следует использовать программу системного обслуживания SYS$SYNCH. Программе системного обслуживания SYS$SYNCH передается флаг события и блок состояния ввода-вывода. Программа SYS$SYNCH ожидает, пока будет установлен флаг события, затем убеждается, что именно программа системного обслуживания, а не какая-либо другая программа установила флаг события, проверяя блок состояния

ввода-вывода. Если блок состояния ввода-вывода все еще нулевой, то программа SYS$SYNCH ожидает, пока заполнится блок состояния ввода-вывода.

Синхронная версия программы системного обслуживания действует точно так, как если бы использовалась асинхронная версия, за которой сразу следует вызов программы SYS$SYNCH. Независимо от того, используется ли синхронная или асинхронная версия программы системного обслуживания, если опускается аргумент EFN, то программа системного обслуживания использует флаг события с нулевым номером.

Пример использования программы системного обслуживания $SYNCH в программе на языке фортран операционной системы МОС ЭП.

Пример.

```
! структура данных для SYS$GETJPI

INTEGER*4 STATUS,
2         FLAG,
2         PID_VALUE
! блок состояния ввода-вывода
INTEGER*2 JPISTATUS,
2         LEN
INTEGER*4 ZERO /0/
COMMON /IO_BLOCK/ JPISTATUS,
2                 LEN,
2                 ZERO
```

```
! вызвать SYS$GETJPI и ждать информацию
STATUS = LIB$GET_EF (FLAG)
IF (.NOT. STATUS) CALL LIB$SIGNAL (%VAL(STATUS))
STATUS = SYS$GETJPI (%VAL(FLAG)),
2                    PID_VALUE,
2                    ,
2                    NAME_BUF_LEN,
2                    JPISTATUS,
2                    ,)
IF (.NOT. STATUS) CALL LIB$SIGNAL (%VAL(STATUS))

STATUS = SYS$SYNCH (%VAL(FLAG),
2                   JPISTATUS)
IF (.NOT. JPISTATUS) THEN
   CALL LIB$SIGNAL (%VAL(JPISTATUS))
END IF
END
```

2.5.2. Режим выполнения процессов

Режимы выполнения процессов оказывают влияние на то, как управление возвращается вызывающей программе, если при выполнении программы системного обслуживания возникает ошибка. Таких режимов два:

1) режим ожидания ресурса;

2) режим исключительной ситуации по ошибке программы системного обслуживания.

Если изменить в программе установку по умолчанию любого из этих режимов, то в результате программа должна будет обрабатывать специальные состояния возврата.

2.5.2.1. Режим ожидания ресурса

Обычно, когда вызывается программа системного обслуживания, а требуемый ресурс недоступен, процесс помещается в состояние ожидания, пока ресурс не станет доступным. После этого программа системного обслуживания завершает свое выполнение. Такой режим называется режимом ожидания ресурса.

Однако, в среде реального времени может быть нежелательным, чтобы программа ждала. В таких случаях можно отключить режим ожидания ресурса, так что если требуемый ресурс недоступен, управление сразу же возвращается в вызывающую программу с кодом состояния, значение которого указывает на ошибку. Режим ожидания ресурса можно отключить (или заново включить) с помощью программы системного обслуживания "установить режим ожидания ресурса (¤SETRWM)".

То, как программа реагирует на недоступность ресурса, в большой степени зависит от программы пользователя и конкретной вызываемой программы системного обслуживания. В некоторых случаях программа имеет возможность продолжить выполнение и попытаться повторить вызов программы системного обслуживания позднее. В других случаях может понадобиться только отметить, что программе потребовалось перейти в ожидание.

2.5.2.2. Режим исключительной ситуации по ошибке программы системного обслуживания

Если при выполнении программы системного обслуживания возникает ошибка, то обычно управление передается следующей инструкции в вызывающей программе, которая может проверить значение возвращенного кода состояния в регистре R0, чтобы определить, успешно или неуспешно прошел вызов программы системного обслуживания.

Чтобы определить ошибку при выполнении вызова программы системного обслуживания, отреагировать на нее, можно использовать механизм обработки кодов состояния операционной системы МОС ЭП. Тогда при возникновении ошибки генерируется исключительное состояние программного обеспечения и управление передается программе обработки кодов состояния.

Данный режим называется режимом исключительной ситуации по ошибке программы системного обслуживания. Он может быть включен (или выключен) с помощью программы системного обслуживания "установить режим исключительной ситуации по ошибке программы системного обслуживания" (¤SETSFM), как показано в примере.

Пример.

```
¤SETSFM_S        ENBFLG=#1
```

Данный вызов включает возможность генерации исключительных ситуаций при возникновении простых или серьезных ошибок во время выполнения программы системного обслуживания (для возвратов с предупреждающими сообщениями исключительные ситуации не генерируются).

Следующие программы системного обслуживания форматирования и преобразования не реагируют на включение режима исключительной ситуации по ошибке и не генерируют соответствующих ситуаций:

- $ASCTIM
- $BINTIM
- $FAO/$FAOL
- $PUTMSG
- $UNWIND

Если пользователь пишет программу, которая должна выполняться с включенным режимом исключительной ситуации по ошибке, то он может написать и подпрограмму обработки кодов состояния. Если не указано пользовательской программы обработки кодов состояния, при возникновении исключительной ситуации, а программа выполняется по команде RUN языка DCL, то программа обработки кодов состояний, принятая по умолчанию, заставляет программу закончить работу и выводит на дисплей описательную информацию о состоянии исключительной ситуации.

Не рекомендуется включать в программах на языках программирования высокого уровня режим исключительной ситуации по ошибке, за исключением, может быть, определенных ситуаций при отладке программы. Если программист включает этот режим в программе, но не об'являет собственную программу обработки кодов состояний, то множество сообщений об ошибках, выводимых на дисплей во время прогона, будут бессмысленными. Компиляторы языков программирования высокого уров-

ня генерируют вызовы программ системного обслуживания для многих предложений или инструкций исходной программы. (Например, чтение и запись файла генерирует вызов системы управления данными СУД-32, которая использует программы системного обслуживания $QIO и $QIOW). Если включен режим исключительной ситуации по ошибке, то много различных типов ошибок (такие как попытка ввода-вывода на несуществующее устройство или нечисловой ввод в математическую программу) будут генерировать одно и то же сообщение "%SYSTEM-F-SSFAIL, исключительная ситуация по ошибке программы системного обслуживания

2.6. Значения кодов состояния, возвращаемых из программ системного обслуживания

Когда программа системного обслуживания заканчивает выполнение, всегда возвращается числовое значение кода состояния. При вызове программ системного обслуживания на языке макроассемблер операционной системы МОС ВП значение кода состояния возвращается в общем регистре R0. Механизмы, используемые в языках программирования высокого уровня, различны.

В зависимости от конкретных нужд, можно проверить только младший бит, три младших бита или все значение целиком:

1) младший бит индицирует успешное (1) или неуспешное (0) завершение программы системного обслуживания;

2) три младших бита, взятые вместе, представляют серьезность ошибки. Значения кодов серьезности ошибки перечислены в табл. 9.

Символические имена определяются макрокомандой определения символических имен ¤STSDEF;

3) остальные биты (с 3 по 31) классифицируют конкретное состояние возврата и компоненту операционной системы МОС ЭП, которая выдала значение кода состояния.

Для значений кодов состояния возврата программ системного обслуживания старшее слово (биты с 16 по 31) содержат нули.

Таблица 9

Значение	Описание	Символическое имя
0	Предупреждение	STS¤K_WARNING
1	Успех	STS¤K_SUCCESS
2	Ошибка	STS¤K_ERROR
3	Информационное сообщение	STS¤K_INFO
4	Серьезная или фатальная ошибка	STS¤K_SEVERR
5-7	Не используются	

Каждое числовое значение кода состояния имеет уникальное символическое имя в следующем формате:

SS¤_код

где код - мнемоника, описывающая состояние возврата.

Примеры:

1. SS$_NORMAL - успешное завершение

2. SS$ACCVIO - нарушение режима доступа

Символические определения для значений кодов состояний включены в системную библиотеку (SYS$LIBRARY:STARLET.OLB), которая принимается по умолчанию. Листинг этих символических имен можно получить во время ассемблирования, вызвав системную макрокоманду $SSDEF. Символические имена значений системных кодов состояния используют для проверки возвращаемых кодов состояния, обработки кодов состояний.

2.6.1. Информация, содержащаяся в кодах состояний

Значения кодов состояния, возвращаемые программами системного обслуживания, могут содержать определенную информацию, т.е. они не всегда только указывают на успешное или неуспешное завершение программы системного обслуживания. Код SS$NORMAL является значением, индицирующим успешное завершение, однако определены также и другие значения. Например, значение кода состояния SS$BUFFEROVF, которое возвращается, если длина строки символов выходных данных программы системного обслуживания больше, чем размер принимающего буфера, является кодом успеха. Однако, это значение дает программе дополнительную информацию.

Возвраты с предупреждающими сообщениями и возвраты с некоторыми ошибками указывают, что программа системного

обслуживания могла выполнить некоторую часть, но не всю запрошенную функцию.

Возможные значения кодов состояния, которые может возвращать каждая программа системного обслуживания, описаны в описаниях отдельных программ (см. раздел 13). Программист, который пишет вызовы программ системного обслуживания, должен прочитать описания значений возвращаемых кодов состояний, чтобы определить, будет ли его программа проверять конкретные коды состояния.

2.6.2. Проверка возвращаемых кодов состояний

Чтобы проверить, успешно ли закончился вызов программы системного обслуживания, программа может проанализировать младший бит регистра R0 и перейти к программе контроля ошибок, если этот бит содержит 0.

Пример 1.

```
BLBC    R0,ERRLABEL
```

где ERRLABEL - метка программы обработки ошибок.

Программы не проверяются на успешное окончание, сравнивая состояние возврата с символическим именем SS$_NORMAL.

Программа контроля ошибок может контролировать специфические значения или специфические уровни серьезности ошибок. Например, следующая инструкция проверяет ошибочное состояние при недопустимом числе флагов событий (пример 2).

Пример 2.

```
    CMPL    #SS$_ILLEFC,R0      допустимо ли число флагов
                                событий
```

Значения кодов состояния возврата всегда имеют размер длинного слова, однако старшие слова всех значений кодов состояния, которые возвращаются в регистре R0 программами системного обслуживания, всегда одинаковые.

2.6.3. Системные сообщения, генерируемые кодами состояния

Если программа выполняется по команде RUN языка DCL, то интерпретатор команд использует содержимое регистра R0 для выдачи описательного сообщения при неуспешном завершении программы.

Следующий фрагмент кода показывает пример программы контроля ошибок в основной программе.

Пример.

```
    $READEF_S-
            EFN=#64,-
            STATE=TEST
    BSBW    ERROR

ERROR:  BLBC    R0,10$      проверить регистр R0
        RSB                 успешно, вернуться
10$:    RET                 выход с кодом состояния в R0
```

Вслед за вызовом программы системного обслуживания инструкция BSBW осуществляет переход к подпрограмме ERROR.

подпрограмма проверяет младший бит в регистре R0 и, если он равен 0, то переходит к инструкции RET, которая приводит к выходу с сохраненным кодом состояния в регистре R0. В противном случае (если младший бит в R0 равен 1) подпрограмма выдает инструкцию RSB для возврата в основную программу.

Если кластер флагов событий, затребованный в данном вызове программы системного обслуживания ¤READEF, не доступен в данный момент процессу, то происходит выход из программы, а интерпретатор команд выдает следующее сообщение:

%SYSTEM-F-UNASEFC кластер флагов событий отсоединен

Ключевое слово UNASEFC в сообщении соответствует значению кода состояния SS¤UNASEFC.

Три серьезные ошибки, приведенные в табл. 10, которые генерируются самими вызовами, а не программами системного обслуживания, могут возвращаться из вызовов программ системного обслуживания.

Таблица 10

Ошибка	Значение
SS¤_ACCVIO	Список аргументов не может быть считан вызывающей программой (используя макрокоманду ¤имя_G) и программа системного обслуживания не вызывается. Это верно не для всех программ системного обслуживания. Если SS¤ACCVIO возвращается из отдельных программ системного обслуживания, то программа

Продолжение табл. 10

Ошибка	Значение
	вызывается, но при этом один или более аргументов программы не могут быть прочитаны или записаны вызывающей программой
SS$_INSFARG	Недостаточно аргументов для вызова программы
SS$_ILLSER	Была вызвана недопустимая программа системного обслуживания

2.7. Вызовы программ системного обслуживания из языков программирования высокого уровня

Каждый язык высокого уровня, который поддерживается операционной системой МОС ВП, обеспечивает механизм для вызова внешней программы и передачи аргументов в эту программу. Специфика этого механизма и используемая терминология зависит от конкретного языка программирования. В рамках данного руководства невозможно описать способы, которыми каждый язык высокого уровня вызывает программы системного обслуживания.

Программы системного обслуживания операционной системы МОС ВП являются внешними программами, которые принимают аргументы. Имеется три способа передачи аргументов программе системного обслуживания (рис. 21 - 23):

1) по значению. Если аргумент размером в длинное слово из списка аргументов содержит реальные данные, которые должны использоваться программой, то реальные данные передаются программе по значению. В этом случае в длинном слове аргумента содержатся реальные данные, другими словами, аргумент и есть реальные данные. Поскольку длина аргумента равна длинному слову, то по значению могут передаваться только такие данные, которые представляются длинным словом;

2) по ссылке. Если аргумент размером в длинное слово из списка аргументов содержит адрес данных, которые должны использоваться программой, то данные передаются по ссылке. В этом случае аргумент является указателем на данные;

3) по дескриптору. Если аргумент размером в длинное слово из списка аргументов содержит адрес дескриптора, то данные передаются по дескриптору. Дескриптор состоит из двух или более длинных слов (в зависимости от типа используемого дескриптора, в которых описываются тип данных, длина и адрес тех данных, которые должны использоваться вызываемой программой; в этом случае аргумент используется как указатель на дескриптор, который в свою очередь является указателем на реальные данные.

В описании каждой программы системного обслуживания (см. раздел 13) указано, как должен передаваться каждый аргумент. Такие фразы как "адрес" и "адрес дескриптора строки символов" указывают, что это аргумент-ссылка или аргумент-дескриптор, соответственно. Фразы типа "булево значение", "число", "значение" или "маска" указывают, что

аргумент передается по значению. Как аргументы передаются программам системного обслуживания (см. рис. 21 - 23).

Для некоторых программ системного обслуживания также требуются структуры данных, специфичные для конкретной программы системного обслуживания, которые указывают, какие должны быть выполнены функции или какая информация должна возвращаться.

2.7.1. Проверка значений возвращаемых кодов состояния в языках программирования высокого уровня

Когда программа системного обслуживания возвращает управление в программу пользователя, она помещает в общий регистр R0 значение состояния возврата. Значение в младшем байте показывает, что либо программа системного обслуживания закончилась успешно, либо какая-то конкретная ошибка не дала выполнить некоторые или все свои функции. После каждого вызова программы системного обслуживания необходимо проверить, закончилась ли она успешно. Можно также проверить состояния, описывающие конкретные ошибки.

Каждый язык обеспечивает определенный механизм для проверки состояний возврата. Часто требуется только проверить младший бит, сравнивая его с логическими значениями TRUE (успешное окончание или информационное сообщение при возврате) и FALSE (ошибка или предупреждающее сообщение при возврате).

Для проверки всего значения на конкретное состояние возврата, каждый язык обеспечивает способ определения в программе значений, связанных с конкретными символически заданными кодами. В программе при кодировании теста на конкретные состояния следует использовать эти символические имена.

В руководствах пользователей по конкретным языкам изложена информация о контроле кодов возврата.

Аргумент, передаваемый по значению

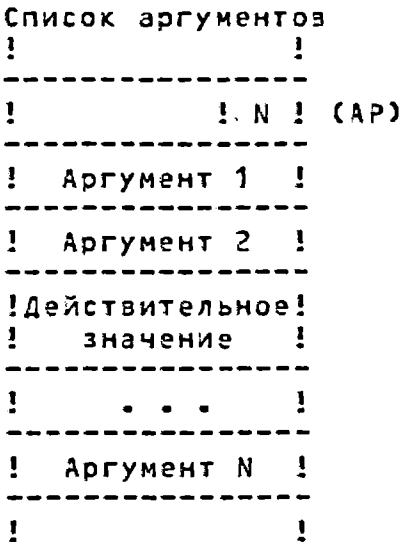


Рис. 21

Аргумент, передаваемый по ссылке

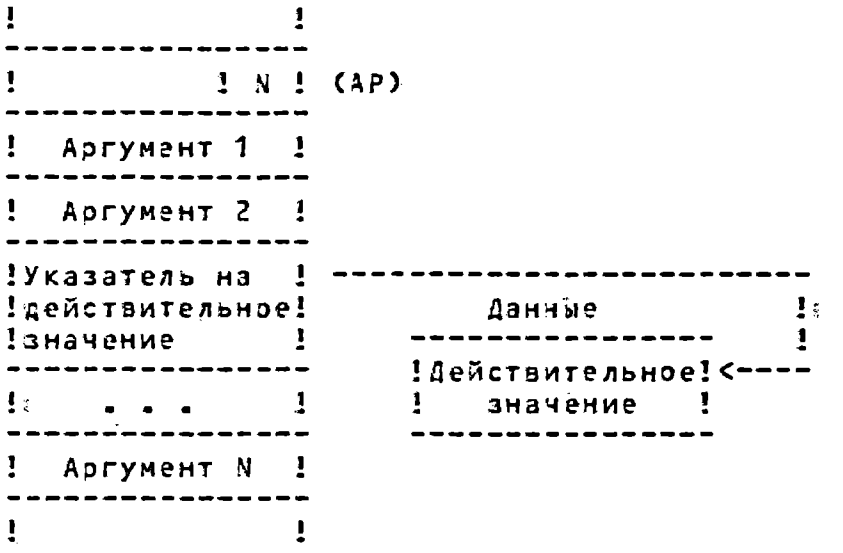


Рис. 22

Аргумент, передаваемый по дескриптору

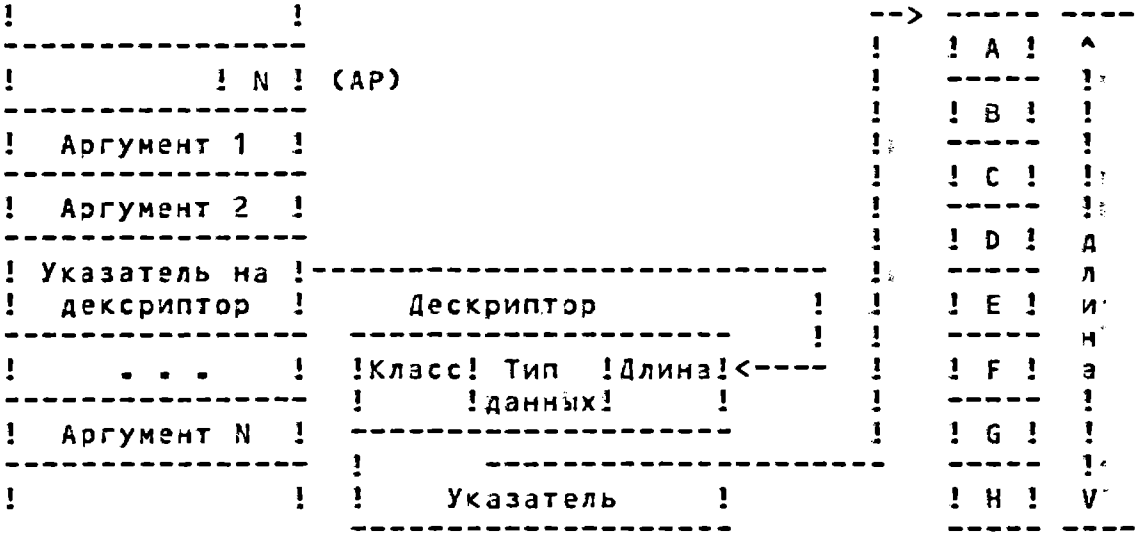


Рис. 23

3. Программы системного обслуживания обеспечения защиты

Программы системного обслуживания обеспечения защиты операционной системы МОС ВП обеспечивают различные механизмы для повышения уровня защиты операционной системы МОС ВП. Программы системного обслуживания включают средства для:

1) создания и сопровождения базы данных прав;

2) создания и трансляции элементов списка управления доступом;

3) модификации списка прав процессов;

4) контроля защиты доступа;

5) обеспечения шаблона стирания секретных данных;

6) управления доступом к магнитным лентам.

В табл.11 перечислены программы системного обслуживания, имеющие отношение к защите операционной системы мос вп.

Таблица 11

Имя программы системного обслуживания	Функция
¤ADD_HOLDER	Добавляет в базу данных прав запись держателя
¤ADD_IDENT	Добавляет идентификатор в базу данных прав

Продолжение табл. 11

Имя программы системного обслуживания	Функция
¤ASCTOID	Транслирует имя идентификатора в двоичное число
¤CHANGE_ACL	Создает или модифицирует список ACL
¤CHKPRO	Вызывает системную программу контроля защиты доступа
¤CREATE_RDB	Инициализирует базу данных прав
¤ERAPAT	Генерирует шаблон стирания секретных данных
¤FIND_HELD	Возвращает идентификатор(ы), принадлежащий держателю в базе данных
¤FIND_HOLDER	Возвращает держатель идентификатора в базе данных прав
¤FINISH_RDB	Отменяет поток записей и очищает значение контекста при поиске в базе данных прав
¤FORMAT_ACL	Форматирует элемет ACL в строку текста
¤GRANTID	Добавляет идентификатор к процессу или в системный список прав
¤IDTOASC	Транслирует значение идентификатора в имя этого идентификатора
¤MOD_HOLDER	Модифицирует запись держателя в базе

Продолжение табл. 11

Имя программы системного обслуживания	Функция
	данных прав
¤MOD_IDENT	Модифицирует запись идентификатора в базе данных прав
¤MTACCESS	Управляет доступом к магнитным лентам
¤PARSE_ACL	Преобразовывает текст элемента ACL в двоичный формат
¤REM_HOLDER	Удаляет запись держателя из списка держателей идентификаторов в базе данных прав
¤REM_IDENT	Удаляет идентификатор и всех держателей этого идентификатора из базы данных прав
¤REVOKID	Удаляет идентификатор из процесса или системного списка прав

3.1. Обзор схемы защиты операционной системы МОС ВП

Основой схемы защиты операционной системы МОС ВП является идентификатор, 32-битовое двоичное число, который представляет процесс в системе. Идентификатор может представлять отдельного пользователя, группу пользователей или

какой-то аспект среды, в которой работает пользователь. Процесс является держателем идентификатора, если идентификатор может представлять этот процесс в операционной системе МОС ВП.

Системная база данных прав является идексным файлом, состоящим из записей идентификаторов и держателей этих идентификаторов в операционной системе МОС ВП. Когда пользователь входит в операционную систему МОС ВП, программа LOGINOUT создает список прав для процесса из соответствующих элементов в базе данных прав. Таким образом список прав процесса содержит все идентификаторы, которые имеет процесс. Процесс может быть держателем нескольких идентификаторов. Список прав процесса становится частью процесса и распространяется на любой создаваемый процесс.

Когда процесс пытается получить доступ к какому-то об'екту в операционной системе МОС ВП, последняя использует список прав для проверки защиты. Она сравнивает идентификаторы в списке прав с атрибутами защиты об'екта и разрешает или запрещает доступ к этому об'екту на основе этого сравнения. Другими словами элеметы списка прав еще не гарантируют доступа, они используются операционной системой МОС ВП для контроля защиты, когда процесс пытается получить доступ к об'екту.

Схема защиты операционной системы МОС ВП обеспечивает защиту с помощью механизма списка управления доступом (ACL). Список управления доступом состоит из элементов (ACE), определяющих тип доступа, который идентификатор

имеет к об'ектам типа файл, устройство или почтовый ящик. Когда процесс пытается получить доступ к об'екту со связанным списком ACL, операционная система МОС ВП разрешает или запрещает доступ, основываясь на том, существует ли точное соответствие в базе данных прав для идентификатора из списка ACL.

3.2. Идентификаторы

Базовой компонентой схемы защиты операционной системы МОС ВП является идентификатор. Это 32-битовое двоичное число представляет различные типы пользователей, использующих операционную систему МОС ВП. Эти типы пользователей включают отдельных пользователей, группы пользователей и различные среды, в которых оперирует процесс.

3.2.1. Форматы идентификаторов

Идентификаторы имеют два формата в базе данных прав: формат UIC и формат ID, представленные на рис. 24 и 25. Старшие биты значения идентификатора задают его формат. Если два старших бита имеют значение нуль, то это указывает на формат UIC идентификатора, если бит 31 установлен, то это указывает на формат ID идентификатора.

Каждый идентификатор UIC является уникальным и представляет пользователя операционной системы МОС ВП. Идентификатор UIC содержит два старших бита, которые определяют формат, поле члена, поле группы. Номера членов меняются от

0 до 65534, номера группы меняются от 1 до 16382.

Формат UIC

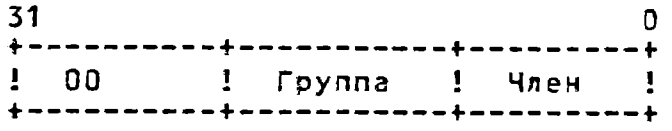


Рис. 24

Если бит 31 установлен, то это указывает на формат ID. Биты с 30 по 28 не используются. Остальные биты задают значение идентификатора.

Формат ID

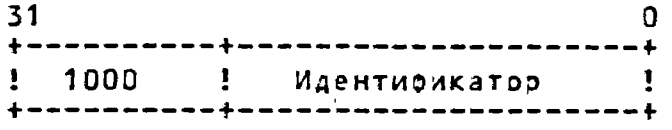


Рис. 25

3.2.2. Имена идентификаторов

Для операционной системы МОС ЗП идентификаторы являются двоичными числами, однако, чтобы сделать идентификаторы более удобными для использования, операционная система МОС ВП транслирует двоичное значение идентификатора в имя идентификатора. Двоичное значение и имя идентификатора связаны в базе данных прав.

Имя идентификатора содержит от 1 до 31 алфавитно-цифровых символа, из которых по крайней мере один символ не цифра. Имя идентификатора не может целиком состоять из цифровых символов. Оно может включать символы от A

до Z, знаки денежной единицы (¤) и подчеркивания (_), а также цифры от 0 до 9. Любой строчный символ автоматически преобразовывается в заглавный.

3.2.3. Идентификаторы, определенные операционной системой МОС ВП

Идентификаторы, определенные операционной системой мос вп, или идентификаторы, задающие среду, автоматически определяются при инициализации базы данных прав. Следующие идентификаторы, определенные операционной системой МОС ВП непосредственно согласуются с классами подключения и имеют отношение к среде, в которой работает процесс:

1) BATCH - все попытки получить доступ, сделанные пакетными заданиями;

2) NETWORK - все попытки получить доступ, сделанные через программу системного обеспечения "трал";

3) INTERACTIVE - все попытки получить доступ, сделанные интерактивными процессами;

4) LOCAL - все попытки получить доступ, сделанные пользователями, подключенными через локальные терминалы;

5) DIALUP - все попытки получить доступ, сделанные пользователями, подключенными к коммутируемым терминалам;

6) REMOTE - все попытки получить доступ, сделанные пользователями, подключенными к сети.

В зависимости от среды, в которой работает процесс, программа LOGINOUT включает один или более из этих идентификаторов при создании списка прав процесса.

3.2.4. Общие идентификаторы

Общие идентификаторы предоставляются пользователям путем создания записей держателей в базе данных прав. Общие идентификаторы могут идентифицировать отдельного пользователя, отдельную группу UIC, группу пользователей или несколько групп.

Идентификаторы и их держатели определяются в базе данных прав с помощью обслуживающей программы авторизации или с помощью соответствующей программы системного обслуживания. Можно также так определить идентификатор в базе данных прав, чтобы позволить пользователям из различных групп UIC иметь этот идентификатор.

Альтернативная группировка, которая описана здесь, позволяет каждому пользователю быть членом многих перекрывающихся групп. Кроме того это позволяет управлять типами доступа более точно, чем с помощью защиты, основанной на

идентификаторах UIC.

Также можно определять идентификаторы таким образом, чтобы они представляли конкретные терминалы, время дня или другие атрибуты, имеющие отношение к среде, специфичной для данной установки (документы [7] и [8]). Такие идентификаторы не получают записей держателей в базе данных прав, но могут быть предоставлены пользователям с помощью привилегированного программного обеспечения, написанного потребителем. Эта особенность системы защиты придает гибкость каждой установке и (поскольку идентификаторы могут быть достаточно специфичны для установки) повысить защиту.

3.2.5. Атрибуты идентификатора

Идентификатор имеет атрибуты, связанные с ним в базе данных прав. Часть списка прав процесса включает в себя атрибуты любого идентификатора, связаного с процессом. Держатель идентификатора может использовать атрибут, только если идентификатор имеет соответствующий атрибут.

В настоящее время единственным реализованным атрибутом является атрибут RESOURCE (ресурс). Атрибут RESOURCE позволяет держателю идентификатора запрашивать ресурсы типа блоков дисковой памяти для идентификатора и, наоборот, держатель, не имеющий атрибута RESOURCE, не может запрашивать ресурсы для идентификатора.

3.3. Базы данных прав

База данных прав является индексным файлом, содержащим два типа записей, которые определяют все идентификаторы: записи идентификаторов и записи держателей.

Для каждого идентификатора в базе данных прав имеется одна запись идентификатора. Запись идентификатора связывает имя идентификатора с 32-битовым двоичным значением и задает атрибуты идентификатора. На рис. 26 показан формат записи идентификатора.

Формат записи идентификатора

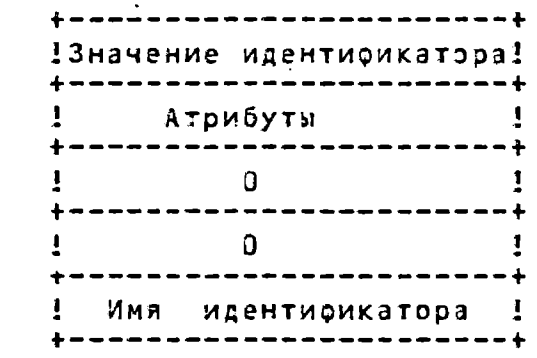


Рис. 26

Для каждого держателя каждого идентификатора в базе данных прав имеется одна запись держателя. Запись держателя связывает держателя с идентификатором, задает атрибуты держателя и указывает идентификатор UIC держателя. Формат записи держателя показан на рис. 27.

Формат записи держателя идентификатора

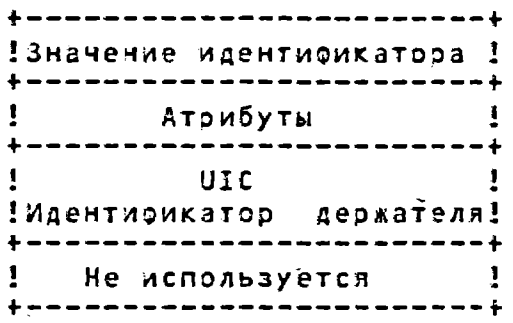


Рис. 27

База данных прав является индексным файлом с тремя ключами. Первичным ключом является значение идентификатора, вторичным ключом является идентификатор держателя и третьим ключом является имя идентификатора. Когда программа LOGINOUT создает список прав процесса, все права этого процесса быстро извлекаются из базы данных прав, используя идентификатор держателя в качестве вторичного ключа.

3.3.1. Инициализация базы данных прав

База данных прав инициализируется одним из следующих способов:

1) при установке или усовершенствовании операционной системы МОС ВП;

2) с помощью обслуживающей программы авторизации;

3) с помощью программы системного обслуживания ¤CREATE_RDB.

При вызове программы системного обслуживания ¤CREATE_RDB можно использовать аргумент SYSID для передачи

значения идентификации операционной системы МОС ВП, связанного с базой данных прав. Если аргумент SYSID опускается, то операционная система МОС ВП использует текущее системное время в 64-битовом формате. Если база данных прав уже существует, то программа системного обслуживания ¤CREATE_RDB завершается с кодом ошибки RMS¤_FEX. Чтобы создать новую базу данных прав, когда она уже существует, необходимо либо явно удалить старую базу данных, либо переименовать ее.

Когда база данных прав инициируется, ей присваивается логическое имя RIGHTSLIST, которое необходимо определить как системное логическое имя в режиме управления. Если логическое имя отсутствует, то база данных прав получает файловую спецификацию, принимаемую по умолчанию, SYS¤SYSTEM:RIGHTSLIST.DAT.

После создания файл RIGHTSLIST.DAT имеет защиту по умолчанию (S:RWED,O:RWED,G:RWE,W:R). Доступ всем пользователям к оглавлению базы данных необходим для того, чтобы все пользователи могли читать записи в базе данных. Чтобы использовать программу системного обслуживания ¤CREATE_RBD, необходим доступ с записью в базу данных. Если база данных находится в группе файлов SYS¤SYSTEM, что принимается по умолчанию, то для получения доступа с записью в базу данных требуется привилегия SYSPRV.

Когда программа системного обслуживания ¤CREATE_RDB инициализирует базу данных прав, автоматически создаются идентификаторы, определенные операционной системой МОС ВП,

описывающие среду, в которой может работать процесс.

Чтобы добавить другие идентификаторы в базу данных прав необходимо определить их с помощью обслуживающей программы авторизации или с помощью соответствующей программы системного обслуживания.

3.3.2. Использование программ системного обслуживания для работы с базой данных прав

Записи идентификатора и держателя в базе данных прав содержат следующие элементы:

1) двоичное значение идентификатора;

2) имя идентификатора;

3) держатель (или держатели) каждого идентификатора;

4) атрибуты каждого идентификатора и каждого держателя каждого идентификатора.

Для того чтобы добавлять, удалять, отображать, модифицировать или транслировать различные элементы базы данных прав используется обслуживающая программа авторизации или одна из программ системного обслуживания, из приведенных в табл. 12.

Таблица 12

Действие	Элемент	Используемая программа системного обслуживания
Транслировать	Имя идентификатора в двоичное значение идентификатора.	¤ASCTOID
	двоичное значение идентификатора в имя идентификатора	¤IDTOASC
Добавить	Запись держателя к идентификатору	¤ADD_HOLDER
	новую запись идентификатора.	¤ADD_IDENT
Найти:	Значение идентификатора, принадлежащего держателю.	¤FIND_HELD
	Держателя(ей) идентификатора.	¤FIND_HOLDER
	Все идентификаторы	¤IDTOASC
Модифицировать	Атрибуты в записи держателя.	¤MOD_HOLDER
	Атрибуты и записи идентификатора	¤MOD_IDENT
Удалить	Держателя из записи	¤REM_HOLDER

Продолжение табл. 12

Действие	Элемент	Используемая программа системного обслуживания
	идентификатора.	
	Идентификатор и всех его держателей	¤REM_IDENT

Необходимые типы доступа при использовании этих программ системного обслуживания приведены в табл. 13. Если база данных прав находится в группе файлов SYS¤SYSTEM, что принимается по умолчанию, то для получения доступа с записью в базу данных требуется привилегия SYSPRV.

Таблица 13

Программа системного обслуживания	Требуемый доступ
¤ADD_HOLDER	Доступ на запись
¤ADD_IDENT	Доступ на запись
¤ASCTOID	Доступ на чтение
¤CREATE_RDB	Доступ на запись
¤FIND_HELD	Доступ на чтение
¤FIND_HOLDER	Доступ на чтение
¤FINISH_RDB	Доступ на чтение
¤IDTOASC	Доступ на чтение

Продолжение табл. 13

Программа системного обслуживания	Требуемый доступ
¤MOD_HOLDER	Доступ на запись
¤MOD_IDENT.	Доступ на запись
¤REM_HOLDER	Доступ на запись
¤REM_IDENT	Доступ на запись

3.3.2.1. Трансляция имен идентификаторов и двоичных значений

Для операционной системы МОС ВП идентификатором является двоичное 32-битовое значение. Чтобы сделать идентификаторы более удобными для использования, двоичное значение имеет связанное с ним имя идентификатора. Значение идентификатора и строка имени идентификатора в коде КОИ-8 связываются в базе данных прав. Для трансляции одного формата в другой можно использовать программы системного обслуживания ¤ASCTOID и ¤IDTOASC. Когда программе ¤ASCTOID передается адрес дескриптора строки, указывающего на имя идентификатора, она возвращает двоичное значение идентификатора. И наоборот, для трансляции двоичного значения идентификатора в строку имени идентификатора в коде КОИ-8 используют программу системного обслуживания ¤IDTOASC.

Программу системного обслуживания ¤IDTOASC можно также использовать для выдачи списка имен всех идентификаторов в

базе данных прав. Для этого необходимо задать аргумент ID равным -1, инициализировать аргумент CONTEXT нулем и повторно вызывать программу ¤IDTOASC до тех пор, пока не будет возвращен код SS¤_NOSUCHID. Программа системного обслуживания ¤IDTOASC возвращает имена идентификаторов в алфавитном порядке. Когда возвращает код SS¤_NOSUCHID, программа ¤IDTOASC очищает длинное слово контекста и отменяет поток записей. Если вызовы программы ¤IDTOASC прекращаются до того, как будет возвращен код SS¤_NOSUCHID, то необходимо использовать программу системного обслуживания ¤FINISH_RDB для очистки длинного слова контекста и отмены потока записей.

3.3.2.2. Добавление идентификаторов и держателей в базу данных прав

Идентификаторы в базу данных прав добавляются с помощью вызова программы системного обслуживания ¤ADD_IDENT. Необходимо использовать аргумент NAME для передачи имени идентификатора, который необходимо добавить к базе данных прав. С помощью аргумента ID можно указать значение идентификатора. Однако, если значение идентификатора не указывается, то операционная система МОС ЭП выбирает это значение из области общих идентификаторов.

Кроме определения значения идентификатора и его имени программа ¤ADD_IDENT может быть использована для указания атрибута RESOURCE в записи идентификатора. Атрибут RESOURCE допустим для держателя идентификатора, только если он уста-

новлен как в записи идентификатора, так и в записи держателя. Аргумент ATTRIB представляет длинное слово, содержащее битовую маску, которая задает атрибуты. Символическое имя KGB¤V_RESOURCE, определенное в системной макробиблиотеке ¤KGBDEF, устанавливает бит RESOURCE в маске. (Вместо префикса KGB¤V можно использовать KGB¤M).

После успешного завершения выполнения программы системного обслуживания ¤ADD_IDENT в базе данных прав будет сформирована новая запись идентификатора, содержащая значение идентификатора, имя идентификатора и атрибуты идентификатора.

После того как запись идентификатора занесена в базу данных прав, можно определить держателя (или держателей) этого идентификатора с помощью программы системного обслуживания ¤ADD_HOLDER. С помощью аргумента ID передается двоичное значение идентификатора, держатель задается аргументом HOLDER, который является адресом квадрослова, содержащего структуру данных, формат которой представлен на рис. 28.

Формат записи держателя

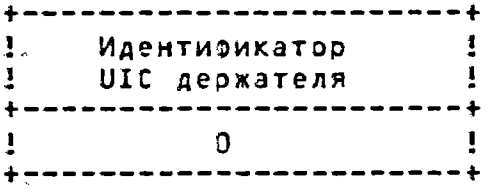


Рис. 28

В базе данных прав идентификатор держателя имеет формат UIC. Атрибуты держателя задаются аргументом ATTRIB точ-

но также как и для программы ¤ADD_IDENT. Атрибут RESOURCE допустим для держателя идентификатора, только если он установлен как в записи идентификатора, так и в записи держателя.

После завершения выполнения программы системного обслуживания ¤ADD_HOLDER в базе данных прав будет сформирована новая запись держателя, содержащая двоичное значение идентификатора, который принадлежит держателю, атрибуты держателя и идентификатор UIC держателя.

3.3.2.3. Определение держателей идентификаторов

Держателя (или держателей) конкретного идентификатора можно определить с помощью вызова программы системного обслуживания ¤FIND_HOLDER. При вызове программы ¤FIND_HOLDER для передачи двоичного значения идентификатора используется аргумент ID. При успешном завершении выполнения программа ¤FIND_HOLDER возвращает идентификатор держателя в аргументе HOLDER и атрибуты держателя в аргументе ATTRIB.

Чтобы найти всех держателей идентификатора, необходимо инициализировать аргумент CONTEXT нулем и последовательно вызывать программу ¤FIND_HOLDER (п. 3.3.3). Поскольку программа системного обслуживания ¤FIND_HOLDER находит записи по одному ключу (идентификатор держателя), то она возвращает записи в том порядке, в котором они были записаны.

3.3.2.4. Определение удерживаемых идентификаторов

Идентификатор (или идентификаторы), удерживаемые держателем, можно определить, вызвав программу системного обслуживания ¤FIND_HELD. При вызове программы ¤FIND_HELD используется аргумент HOLDER для указания того держателя, чей идентификатор необходимо найти.

При завершении выполнения программа ¤FIND_HILD возвращает двоичное значение идентификатора и атрибуты.

Можно найти все идентификаторы, удерживаемые указанным держателем, если инициализировать нулем аргумент CONTEXT и последовательно вызывать программу ¤FIND_HELD (см. П. 3.3.3). Поскольку программа ¤FIND_HELD находит записи по одному ключу (идентификатор), то она возвращает записи в том порядке, в котором они были записаны.

3.3.2.5. Модификация записи идентификатора

С помощью вызова программы системного обслуживания ¤MOD_IDENT можно модифицировать имя идентификатора, его значение и/или атрибут в базе данных прав. Для передачи двоичного значения модифицируемого идентификатора используется аргумент ID. Чтобы установить атрибут, необходимо использовать аргумент SET_ATTRIB, являющийся длинным словом, содержащим битовую маску, которая задает атрибут. Символическое имя KGB¤V_RESOURCE, определенное в системной макробиблиотеке ¤KGBDEF, используют, чтобы установить бит

RESOURCE в битовой маске атрибута. (Вместо префикса KGB¤V можно использовать префикс KGB¤M).

Если атрибут не требуется для идентификатора, то используется аргумент CLR_ATTRIB, являющийся длинным словом, содержащим битовую маску, которая задает атрибут. Если один и тот же атрибут задается в аргументе SET_ATTRIB и в аргументе CLR_ATTRIB, то атрибут устанавливается.

С помощью аргументов NEW_NAME и NEW_VALUE можно изменить имя идентификатора и/или его значение. Аргумент NEW_NAME является адресом дескриптора, указывающего на строку имени идентификатора. Аргумент NEW_VALUE является длинным словом, содержащим двоичное значение идентификатора. Если изменяется значение идентификатора, который является держателем других идентификаторов (например, UIC), то программа ¤MOD_IDENT обновляет все существующие записи держателей новым значением идентификатора держателя.

После успешного завершения выполнения программы ¤MOD_IDENT, в базе данных прав существует новая запись идентификатора, содержащая значение идентификатора, его имя и атрибуты.

3.3.2.6. Модификация записи держателя

Вызывая программу системного обслуживания ¤MOD_HOLDER, можно модифицировать запись держателя. При вызове программы ¤MOD_HOLDER аргументы ID и HOLDER используется для передачи двоичного значения идентификатора и идентификатора UIC держателя, запись которого требуется модифицировать.

Программу ¤MOD_HOLDER можно использовать для установки и отключения атрибутов идентификатора точно так же, как и при работе с программой ¤MOD_IDENT.

После завершения работы программы ¤MOD_HOLDER в базе данных прав будет сформирована новая запись держателя, содержащая значение идентификатора, идентификатор держателя и атрибуты держателя.

3.3.2.7. Удаление идентификаторов и держателей из базы данных прав

Для удаления идентификатора и его держателей из базы данных прав используется программа системного обслуживания ¤REM_IDENT. При вызове программы ¤REM_IDENT аргумент ID используется для передачи двоичного значения идентификатора, который необходимо удалить. После завершения выполнения программы ¤REM_IDENT идентификатор и все связанные с ним записи держателей удаляются из базы данных прав.

Чтобы удалить держателя из списка держателей идентификатора, используют программу системного обслуживания ¤REM_HOLDER. При вызове программы ¤REM_HOLDER аргументы ID и HOLDER используются для передачи двоичного значения идентификатора и идентификатора UIC держателя, запись которого требуется удалить.

При успешном завершении выполнения программа ¤REM_HOLDER удаляет держателя из списка держателей идентификатора.

3.3.3. Операции поиска

При использовании программ системного обслуживания ¤IDTOASC, ¤FIND_HELD и ¤FIND_HOLDER можно выполнять поиск во всей базе данных прав. Для этого необходимо инициализировать длинное слово контекста нулем и повторять вызов одной из этих трех программ системного обслуживания, пока не будет возвращен код состояния SS¤NOSUCHID. После возврата этого кода программа системного обслуживания чистит длинное слово контекста и отменяет поток записей. Если закончить вызовы одной из этих программ системного обслуживания до того, как будет возвращен код SS¤NOSUCHID, то необходимо использовать программу ¤FINISH_RDB для очистки длинного слова контекста и отмены потока записей.

Структура базы данных прав влияет на порядок, в котором каждая из этих программ возвращает записи при поиске в базе данных прав. База данных прав является идексным файлом с тремя ключами. Первичным ключом является двоичное значение идентификатора, вторичным ключом - идентификатор UIC держателя и третьим ключом является имя идентификатора.

Во время операции поиска программа системного обслуживания получает первую запись с помощью индексной операции GET системы управления данными. Ключ, используемый для операции GET, зависит от программы системного обслуживания. Программа ¤FIND_HOLDER использует двоичное значение идентификатора; программа ¤FIND_HELD использует идентификатор UIC держателя. После индексной операции GET программа системного обслуживания возвращает записи с помощью последователь-

ной операции GET системы управления данными. Следовательно, организация файла, ключ, используемый для первой операции GET, и порядок, в котором записи были первоначально записаны в базу данных, определяют, как программа системного обслуживания возвращает записи при операции поиска.

Примеры:

1. ¤IDTOASC — В порядке имен идентификаторов.
2. ¤FIND_HELD — Первая операция GET-ключ держателя. Последующие записи возвращаются в том порядке, в котором они были записаны.
3. ¤FIND_HOLDER — Первая операция GET-ключ идентификатора. Последующие записи возвращаются в том порядке, в котором они были записаны.

3.4. Создание, трансляция и сопровождение элементов ACE

Список управления доступом (ACL) является списком элементов, определяющих тип доступа к об'ектам операционной системы МОС ВП типа файла, устройства или почтового ящика. Когда процесс пытается получить доступ к об'екту со связанным списком ACL, операционная система МОС ВП разрешает доступ на основе типа доступа, заданного элементами в списке ACL.

Элементы списка управления доступом (ACE) имеют двоичную форму. Однако для удобства использования этих ACE они

имеют также формат строки текста. Программы системного обслуживания ¤FORMAT_ACL и ¤PARSE_ACL используются для трансляции элементов ACE из одного формата в другой точно так же, как программы ¤IDTOASC и ¤ASCTOID транслируют идентификаторы из двоичной формы в текстовый формат, из текстового формата в двоичный.

Создавать и обрабатывать списки ACL можно с помощью редактора ACL, команды SET ACL языка DCL и вызова программы системного обслуживания ¤CHANGE_ACL.

3.4.1. Формат элементов ACE

Имеется четыре типа элементов ACE:

1) сигнальный;

2) зависящий от программы пользователя;

3) защиты по умолчанию;

4) идентификатор.

Сигнальный элемент ACE обеспечивает сигнал защиты, когда доступ к об'екту осуществляется специальным образом. Элемент ACE пользователя содержит информацию, зависящую от программы пользователя, или информацию заданную пользователем. Элемент ACE защиты по умолчанию определяет защиту по умолчанию для каталога, так что защита может быть распространена на файлы и подкаталоги, созданные в этом каталоге. Элемент ACE идентификатора управляет типом доступа, разрешенного пользователю или группе пользователей, в соответствии с идентификатором.

Тип элемента ACE определяет его формат. Символические имена, задающие смещения в байтах и значения типов, определены в системной макробиблиотеке (=ACEDEF).

3.4.1.1. Сигнальный элемент ACE

Сигнальный элемент ACE устанавливает сигнал защиты на об'екте операционной системы МОС ВП. На рис. 29 приведен формат сигнального элемента ACE. Описание структуры сигнального элемента ACE приведено в табл. 14.

Формат сигнального элемента ACE

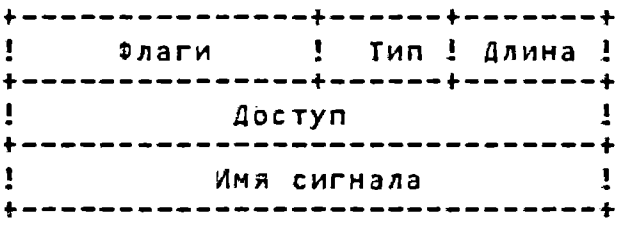


Рис. 29

Таблица 14

Поле	Символическое имя	Описание
Длина	ACE=B_SIZE	Байт, содержащий длину в байтах буфера элемента ACE
Тип	ACE=B_TYPE	Байт, содержащий значение типа ACE=C_ALARM
Флаги	ACE=W_FLAGS	Слово, содержащее информацию о сигнальном элементе ACE и информацию, не зави-

Продолжение табл. 14

Поле	Символическое имя	Описание
		сящую от типа элемента ACE
Доступ	ACE$L_ACCESS	Длинное слово, содержащее маску, которая указывает, какие режимы доступа должны отслеживаться
Имя сигнала	ACE$T_AUDITNAME	Строка символов, содержащая имя сигнала

Слово флагов содержит информацию, специфичную для сигнальных элементов ACE, и информацию, относящуюся ко всем типам элементов ACE. В слове флагов первый байт содержит флаги, специфичные для каждого типа элемента ACE, второй байт содержит флаги общие для всех типов элементов ACE. Символические имена, приведенные в табл. 15 являются битовыми смещениями в слове флагов к информации о сигнальном элементе ACE.

Таблица 15

Бит	Значение, если бит установлен
ACE$V_SUCCESS	Указывает, что сигнал установлен при успешном доступе
ACE$V_FAILURE	Указывает, что сигнал установлен при неудачном доступе

Символические имена, приведенные в табл. 16 являются битовыми смещениями в слове флагов относительно информации, имеющей отношение ко всем типам элементов ACE.

Таблица 16

Бит	Значение, если бит установлен
ACE$V_DEFAULT	Данный элемент ACE добавляется к списку ACL для любого файла, созданного в каталоге, чей список ACL содержит данный элемент ACE. Описанная операция относится только к элементу ACE из списка ACL файла каталога
ACE$V_HIDDEN	Данный элемент зависит от программы пользователя. Команды ACL языка DCL и редактор ACL не могут быть использованы для изменения установки этого бита. команда DIRECTORY/ACL языка DCL не отображает этот бит
ACE$V_NOPROPAGATE	Данный элемент ACE не распространяется на версии файла
ACE$V_PROTECTED	Данный элемент ACE не удаляется, когда удаляется весь список ACL. Этот элемент ACE необходимо удалять только явно

Символические значения, приведенные в табл. 17, являются смещениями в битах в маске доступа. Можно также

получить символические значения в виде масок с соответствующим набором битов, если использовать вместо префикса ACE$V префикс ACE$M.

Таблица 17

Бит	Значение, если бит установлен
ACE$V_READ	Управление доступом на чтение
ACE$V_WRITE	Управление доступом на запись
ACE$V_EXECUTE	Управление доступом на выполнение
ACE$V_DELETE	Управление доступом на удаление
ACE$V_CONTROL	Управление доступом на модификацию поля

3.4.1.2. Элемент ACE пользователя

Элемент ACE пользователя содержит информацию, зависящую от программы пользователя. На рис. 30 показан формат элемента пользователя.

Формат элемента пользователя

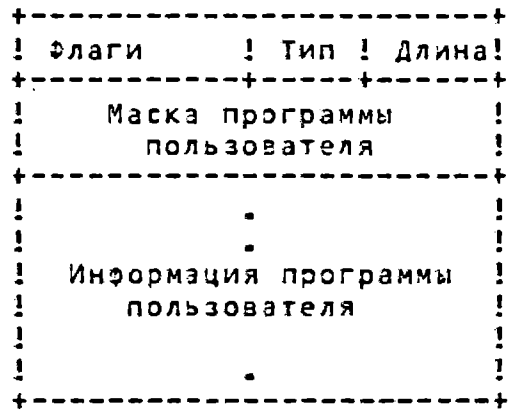


Рис. 30

Символические имена, приведенные в табл. 18 задают смещения в байтах.

Таблица 18

Поле	Символическое имя	Описание
Длина	ACE$B_SIZE	Байт, содержащий длину в байтах буфера элемента ACE
Тип	ACE$B_TYPE	Байт, содержащий значение типа ACE$C_INFO
Флаги	ACE$W_FLAGS	Слово, содержащее информацию элемента ACE пользователя и информацию, не зависящую от типа элемента ACE
Маска	ACE$L_INFO_FLAGS	Длинное слово, содержащее

Продолжение табл. 18

Поле	Символическое имя	Описание
прог- раммы пользо- вателя		маску, которая определена и используется программой пользователя
Инфор- мация прог- раммы пользо- вателя	ACEUT_INFO_START	Структура данных переменной длины, определенная и используемая программой пользователя. Длина этих данных записана в поле длины

Слово флагов содержит информацию, специфичную для элементов ACE пользователя, и информацию, относящуюся ко всем типам элементов ACE. В слове флагов первый байт содержит флаги, специфические для каждого типа элемента ACE, второй байт содержит флаги общие для всех типов элементов ACE. Подробно информация, не зависящая от типа элемента ACE, описана в табл. 16. Символическое имя, приведенное в табл. 19, является смещением в битах в слове флагов к информации элемента ACE пользователя.

Таблица 19

Бит	Значение, если бит установлен
ACE$V_INFO_TYPE	Четырехбитовое поле, содержащее значение, которое указывает, является ли программа пользователя программой типа CSS (ACE$C_CSS) или программой потребителя (ACE$C_CUST)

3.4.1.3. Элемент ACE защиты по умолчанию

Элемент ACE защиты по умолчанию специфицирует защиту по умолчанию всех файлов и подкаталогов, созданных в каталоге. Данный тип элемента ACE можно использовать только в списке ACL файла каталога. На рис. 31 приведен формат этого элемента.

Формат элемента ACE защиты по умолчанию

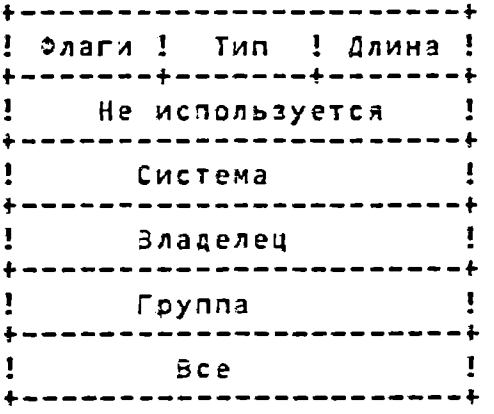


Рис. 31

Символические имена, приведенные в табл. 20 задают смещения в байтах.

Таблица 20

Поле	Символическое имя	Описание
Длина	ACE$B_SIZE	Байт, содержащий длину в байтах буфера элемента ACE
Тип	ACE$B_TYPE	Байт, содержащий значение типа ACE$C_DIRDEF
Флаги	ACE$W_FLAGS	Слово, содержащее информацию, независящую от типа ACE
Не используется	ACE$L_SPARE1	Длинное слово. Должно быть нулевым
Система	ACE$L_SYS_PROT	Длинное слово, содержащее маску, которая индицирует режим доступа, предоставленный системным пользователям. Каждый бит представляет один тип доступа
Владелец	ACE$L_OWN_PROT	Длинное слово, содержащее маску, которая индицирует режим доступа, предоставленный владельцу. Каждый бит представляет один тип доступа

Продолжение табл. 20

Поле	Символическое имя	Описание
Группа	ACE$L_GRP_PROT	Длинное слово, содержащее маску, которая индицирует режим доступа, предоставленный группе пользователей. Каждый бит предствляет один тип доступа
Все	ACE$L_WOR_PROT	Длинное слово, содержащее маску, которая индицирует режим доступа, предоставленный всем пользователям. каждый бит представляет один тип доступа

В слове флагов первый байт содержит флаги специфичные для каждого типа элементов ACE, второй байт содержит флаги общие для всех типов элементов ACE (см. табл. 16).

Операционная система МОС ВП интерпретирует биты в маске доступа, как показано в табл. 21. Символические значения являются смещениями в битах внутри маски, индицирующими режим доступа, предоставленный в полях системы, владельца, группы и всех пользователей.

Таблица 21

Бит	Значение, если бит установлен
ACE$V_READ	Управление доступом на чтение
ACE$V_WRITE	Управление доступом на запись
ACE$V_EXECUTE	Управление доступом на выполнение
ACE$V_DELETE	Управление доступом на удаление

Можно также получить символические значения в виде масок с соответствующим набором битов, используя префикс ACE$M вместо префикса ACE$V.

3.4.1.4. Элемент ACE идентификатора

Элемент ACE идентификатора управляет типом разрешенного доступа на базе идентификаторов. Доступ управляется на основе того, есть или нет точное соответствие между идентификатором списка прав процесса и идентификатором в элементе ACE. Рис. 32 иллюстрирует формат этого элемента.

Формат элемента ACE идентификатора

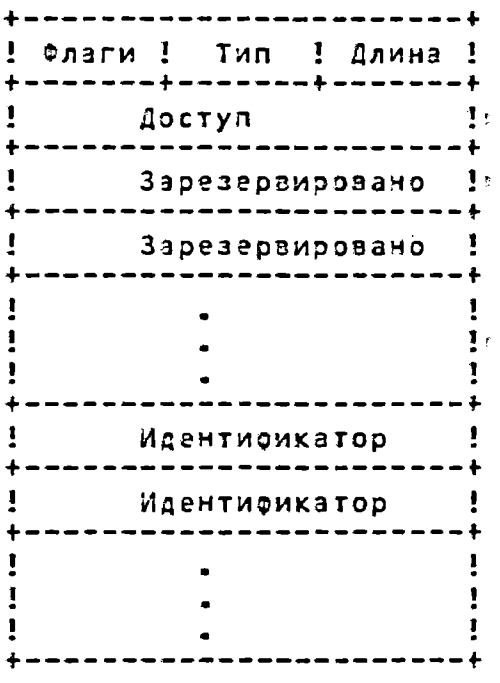


Рис. 32

Символические имена, приведенные в табл. 22 задают смещения в байтах.

Таблица 22

Поле	Символическое имя	Описание
Длина	ACE$B_SIZE	Байт, содержащий длину в байтах буфера элемента ACE
Тип	ACE$B_TYPE	Байт, содержащий значение типа ACE$C_KEYID
Флаги	ACE$W_FLAGS	Слово, содержащее информацию элементА ACE идентифи-

Продолжение табл. 22

Поле	Символическое имя	Описание
		фикатора и информацию, не зависящую от типа элемента ACE
Доступ	ACE$L_ACCESS	Длинное слово, содержащее маску, которая индицирует режим доступа, предоставленный указанным идентификатора
Зарезервировано	ACE$V_RESERVED	Длинные слова, содержащие информацию, относящуюся к программе пользователя. Число зарезервированных слов задается в поле флагов
Идентификатор	ACE$L_KEY	Длинные слова, содержащие идентификаторы. Число этих длинных слов хранится в поле ACE$B_LENGTH. Если пользователь, пытающийся получить доступ, имеет все идентификаторы в списке, то элемент ACE соответствует этому пользователю, и

Продолжение табл. 22

Поле	Символическое имя	Описание
		доступ, указанный в поле ACE$L_ACCESS, предоставляется

Слово флагов содержит информацию, специфичную для элемента ACE идентификатора, и информацию, относящуюся ко всем типам элементов ACE. В слове флагов первый байт содержит флаги, специфичные для каждого типа элементов ACE, второй байт содержит флаги общие для всех типов элементов ACE (см. табл. 16). Символическое имя, приведенное в табл. 23 является битовым смещением к информации элемента ACE идентификатора.

Таблица 23

Бит	Значение, если бит установлен
ACE$V_RESERVED	Четырехбитовое поле, содержащее число длинных слов, зарезервированных для данных, зависящих от программы пользователя. Это число может быть между 0 и 15. Зарезервированные длинные слова, если они имеются непосредственно предшествуют идентификаторам

Символические имена, приведенные в табл. 24, являются битовыми смещениями внутри маски, идинцирующей режим доступа, разрешенный в полях системы, владельца, группы и всех пользователей.

Таблица 24

Бит	Значение, если бит установлен
ACE$V_READ	Управление доступом на чтение
ACE$V_WRITE	Управление доступом на запись
ACE$V_EXECUTE	Управление доступом на выполнение
ACE$V_DELETE	Управление доступом на удаление
ACE$V_CONTROL	Управление доступом на модификацию поля

Символические значения можно также получить в виде масок с соответствующим набором битов, если использовать префикс ACE$M вместо префикса ACE$V.

3.4.2. Трансляция элементов ACE

Для трансляции элементов ACE из двоичного формата в текстовую строку можно использовать программу системного обслуживания $FORMAT_ACL. Аргумент ACLENT является адресом дескриптора, указывающим на буфер, который содержит описание элемента ACE. Первый байт буфера содержит длину элемента ACE, второй байт содержит тип, который в свою очередь определяет формат элемента ACE (табл. 25).

Таблица 25

Символическое имя	Тип элемента
ACE$C_ALARM	Сигнальный элемент ACE
ACE$C_INFO	Элемент ACE пользователя
ACE$C_DIRDEF	Элемент ACE защиты по умолчанию
ACE$C_KEYID	Элемент ACE идентификатора

Аргумент ACLLEN задает длину текстовой строки, записанной в буфер, на которую указывает аргумент ACLSTR. Аргументы WIDTH, TRMDSC и IDENT используются для указания конкретной величины, конечного символа и числа пробелов для элемента ACE. Аргумент ACCNAM содержит адрес массива из 32 квадрослов дескрипторов, которые определяют имена битов в маске доступа элемента ACE.

Если аргумент ACCNAM опущен, то используются следующие имена:

бит 0 - читать;

бит 1 - писать;

бит 2 - выполнить;

бит 3 - удалить;

бит 4 - управлять (в данном случае модифицировать);

бит 5 - BIT_5;

бит 6 - BIT_6;

.

бит 31 - BIT_31.

Программа системного обслуживания $PARSE_ACL транслирует элементы ACE из формата текстовой строки в двоичный формат. Аргумент ACLSTR является адресом дескриптора строки, указывающим на текстовую строку элемента ACE. Так же как и для программы $FORMAT_ACL аргумент ACLENT является адресом дескриптора, указывающего на буфер, содержащий описание элемента ACE. Первый байт буфера содержит длину элемента ACE, второй байт содержит тип, который в свою очередь определяет формат элемента ACE. Если программа $PARSE_ACL заканчивается по ошибке, то аргумент ERRPOS указывает на ошибочную позицию в строке. Аргумент ACCNAM содержит адрес массива из 32 квадрослов дескрипторов, которые определяют имена битов в маске доступа элемента ACE.

3.4.3. Создание и обслуживание элементов ACE

Программа системного обслуживания $CHANGE_ACL используется для создания или модификации списка ACL, связанного с об'ектом. Об'ект, чей список ACL должен модифицироваться, задается либо аргументом CHAN, который указывает канал ввода-вывода, связанного с об'ектом, либо аргументом OBJNAM, который указывает имя об'екта. Если задается аргумент OBJNAM, то аргумент CHAN должен быть либо опущен, либо задан нулем. Аргумент OBJTYP указывает тип об'екта.

Следующие значения задают тип об'екта:

ACL$C_DEVICE - об'ект есть устройство;

ACL$C_FILE - об'ект есть файл;

ACL$C_GROUP_GLOBAL_SECTION - об'ект есть групповая глобальная секция;

ACL$C_LOGICAL_NAME_SECTION - об'ект есть таблица логических имен;

ACL$C_SYSTEM_GLOBAL_SECTION - об'ект есть системная глобальная секция.

Аргумент ACMODE используется для задания режима доступа, который используется при контроле защиты доступа к файлу. По умолчанию используется режим ядра, однако операционная система МОС ВП сравнивает аргумент ACMODE с режимом досула вызывающей программы и использует наименее привилегированный режим. Аргумент ITMLST является списком элементных кодов, задающим изменения, которые необходимо сделать в списке ACL. Каждый элементный код состоит из трех элементов. Рис. 33 иллюстрирует формат элементного кода.

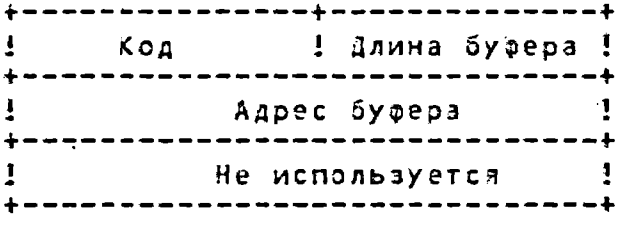


Рис. 33

Элементный список заканчивается длинным словом содержащим нулевое значение. Поле "длина буфера" содержит количество байтов в буфере, содержащем информацию, которая

передается программе или из программы $CHANGE_ACL. Адрес буфера содержится в поле "адрес буфера". Третье длинное слово стандартного элементного дескриптора не используется программой $CHANGE_ACL и должно быть нулевым.

Элементный код специфицирует изменения, которые должны быть сделаны в списке ACL. Следующие символические имена для элементных кодов определены в системной макробиблиотеке ($ACLDEF):

ACL$C_ACLLENGTH - возвращает размер в байтах списка ACL об'екта. Поле "адрес буфера" указывает на длинное слово, содержащее размер;

ACL$C_ADDACLENT - добавляет элемент ACE в начало списка ACL, если аргумент CONTXT равен 0, в конец ACL, если аргумент CONTXT равен минус 1, или в то место, на которое указывает предыдущий код ACL$C_FNDACETYP или ACL$C_EFDACLENT. Поле "адрес буфера" указывает на структуру данных переменной длины, содержащую добавляемый элемент ACE. Используя код ACL$C_ADDaCLENT, можно добавить более одного элемента ACE, при этом поле "адрес буфера" должно содержать общий размер всех элементов ACE, которые находятся в буфере;

ACL$C_DELACLENT - удаляет элемент ACE, на который указывает поле "адрес буфера" или, если

это поле равно нулю, тот элемент ACE, на который указывают предыдущие коды ACL␣C_FNDACETYP или ACL␣_FNDACLENT;

ACL␣C_DELETEACL - удаляет весь список ACL за исключением защищенных элементов ACE;

ACL␣C_FNDACETYP - определяет местоположение элемента ACE того типа, на который указывает поле "адрес буфера";

ACL␣C_FNDACLENT - определяет местоположение элемента ACE, на который указывает поле "адрес буфера";

ACL␣C_RLOCK_ACL - обеспечивает блокировку чтения об'екта, чтобы заблокировать все записи в список ACL об'екта. Независимо от режима вызывающей программы блокировки списка ACL являются блокировками режима пользователя, так что все режимы доступа будут корректно блокировать списки ACL;

ACL␣C_WLOCK_ACL - обеспечивает исключающую блокировку об'екта, чтобы заблокировать все остальные попытки чтения и записи списка ACL об'екта. Независимо от режима вызывающей программы блокировки списка ACL являются блокировками режима пользователя, так что

все режимы доступа будут корректно блокировать списки ACL;

ACL$C_MODACLENT - заменяет элемент ACE, на который указывает предыдущий код ACL$C_FNDACETYP или ACL$C_FNDACLENT, элементом ACE, на который указывает поле "адрес буфера";

ACL$C_READACE - читает элемент ACE, на который указывает код ACL$C_FNDACETYP или ACL$C_FNDACLENT в буфер, на который указывает поле "адрес буфера";

ACL$C_READACL - читает список ACL об'екта. Аргумент CONTXT должен быть первоначально установлен равным нулю. Элементы ACE считываются в буфер, на который указывает поле "адрес буфера";

ACL$C_UNLOCK_ACL - освобождает блокировку, полученную кодами ACL$C_RLOCK_ACL или ACL$C_WLOCK_ACL.

При добавлении элемента ACE кодом ACL$C_ADDACLENT или определении местоположения элемента ACE кодом ACL$C_FNDACETYP или ACL$C_FNDACLENT программа $CHANGE_ACL просматривает список ACL в поисках элемента, соответствующего элементу в буфере ACE. Программа $CHANGE_ACL не всегда находит соответствие, основываясь на содержимом всего буфера ACE. То, как программа находит соответствие, определяется типом элемента ACE. Существуют следующие алгоритмы поис-

ка элементов ACE:

1) элементы ACE защиты по умолчанию (ACE$C_DIRDEF) сопоставляются только по полю типа (ACE$B_TYPE). Поскольку список ACL может иметь только один элемент ACE защиты по умолчанию, то программа $CHANGE_ACL прекращает просмотр, как только обнаружит соответствие типов;

2) элементы ACE идентификатора (ACE$C_KEYID) сопоставляются по полям флагов (ACE$W_FLAGS) и идентификатора (ACE$L_KEY);

3) сигнальные элементы ACE (ACE$C_ALARM) сопоставляются по полям флагов (ACE$W_FLAGS) и маски доступа (ACE$L_ACCESS);

4) все остальные типы элементов ACE сравниваются по полному содержанию буфера ACE.

Поскольку программа $CHANGE_ACL использует правила соответствия, добавление элемента ACE может привести к замене другого элемента ACE. Например, если добавляется элемент ACE идентификатора с теми же полями флагов и идентификатора, но с другой маской доступа, то новый элемент ACE заменит старый. Если добавляется элемент ACE в начало списка ACL, программа $CHANGE_ACL удаляет такой же элемент ACE, т.к. он не может быть найден.

- Если добавляется элемент элемент ACE ниже соответствующего ему элемента в списке ACL, то добавляемый элемент ACE не имеет никакого влияния, поскольку он никогда не будет найден.

3.5. Модификация списка прав

Когда создается процесс, программа LOGINOUT строит список прав для процесса, состоящий из идентификаторов, которые принадлежат пользователю и соответствующих идентификаторов, связанных со средой. Системный список прав, создаваемый по умолчанию, используется в дополнение к любому списку прав процесса. Модификации системного списка прав вызывают модификации прав каждого процесса.

Привилегированная подсистема может изменить системный список прав или список прав процесса с помощью программ системного обслуживания ¤GRANTID или ¤REVOKID. Эти программы системного обслуживания не предназначены для обычного пользователя операционной системы МОС ЭП. Программа ¤GRANTID добавляет идентификатор к списку прав или, если идентификатор уже является частью списка прав, она модифицирует атрибут идентификатора. Программа ¤REVOKID удаляет идентификатор из списка прав. Если идентификатор, указанный аргументами ID или NAME, является владельцем других идентификаторов, то идентификатор удаляется из этих записей держателей.

Программы ¤GRANTID и ¤REVOKID используют аргументы PIDADR и PRCNAM таким же способом, что и все другие программы управления процессами (см. документы [7] и [8]).

3.6. Контроль защиты доступа

Программа системного обслуживания $CHKPRO обращается к контролю защиты доступа, который используется операционной системой МОС ВП. Программа системного обслуживания не разрешает и не запрещает доступ, а только выполняет контроль защиты для программного продукта, прикладной программы и других подобных подсистем, которые в свою очередь должны разрешать или запрещать доступ.

Входная и выходная информация передается программе $CHKPRO с помощью аргумента ITMLST, который является адресом элементного списка дескрипторов. Программа $CHKPRO сравнивает элементный список прав и привилегий пользователя, пытающегося получить доступ к об'екту, со списком атрибутов защиты данного об'екта. Если пользователь может получить доступ к об'екту, то программа $CHKPRO возвращает состояние SS$_NORMAL, в противном случае возвращается состояние SS$_NOPRIV.

Программа $CHKPRO не запрещает и не разрешает доступа. Подсистема сама должна разрешить или запретить доступ, основываясь на выходной информации программы $CHKPRO (коды SS$_NORMAL или SS$_NOPRIV).

Кроме того программа $CHKPRO возвращает элементный список прав и привилегий, которые разрешают пользователю доступ к об'екту, а также имена сигналов защиты, которые подаются при попытке доступа.

3.7. Дополнительные программы системного обеспечения защиты

Операционная система МОС ВП предоставляет две дополнительные программы системного обслуживания, которые оказывают влияние на защиту системы. Служебная программа ¤ERAPAT обеспечивает механизм, посредством которого пользователь может написать шаблон стирания секретных данных для дисков. Шаблон стирания секретных данных может иметь конфигурацию, удовлетворяющую конкретным требованиям установки. Служебная программа ¤MTACCESS проверяет поле доступности в метке магнитной ленты, чтобы определить защищен ли данный том операционной системой МОС ВП.

4. Программы обслуживания флагов событий

Флаги событий являются битами извещения о состоянии, которые обслуживаются операционной системой MOC ЭП и предназначены для общего использования в программах. Программы могут использовать флаги событий для выполнения множества функций сигнализации. Программы обслуживания флагов событий сбрасывают, устанавливают и считывают флаги событий. Они также могут переводить процесс в состояние ожидания, отложив установку флага или флагов событий. Следующие программы системного обслуживания обслуживают флаги событий:

1) подсоединить кластер общих флагов событий (¤ASCEFC);

2) отсоединить кластер общих флагов событий (¤DACEFC);

3) удалить кластер общих флагов событий (¤DLCEFC);

4) установить флаг события (¤SETEF);

5) сбросить флаг события (¤CLREF);

6) прочитать флаги событий (¤READEF);

7) ожидать одного флага события (¤WAITFR);

8) ожидать логического "или" флагов событий (¤WFLOR);

9) ожидать логического "и" флагов событий (¤WFLAND).

Некоторые программы системного обслуживания устанавливают флаг события, чтобы отметить, что событие завершилось или произошло. Вызывающая программа может проверить флаг. Вот некоторые из программ системного обслуживания, которые используют флаги событий, чтобы сигнализировать о событиях вызывающему процессу:

1) поставить в очередь запрос на блокирование (¤ENQ и ¤ENQW);

2) получить информацию об устройстве или томе (¤GETDVI и ¤GETDVIW);

3) получить информацию о задании или процессе (¤GETJPI и ¤GETJPIW);

4) получить информацию обо всей системе (¤GETSYI и ¤GETSYIW);

5) поставить в очередь запрос на ввод-вывод (¤QIO и ¤QIOW);

6) установить таймер (¤SETIMR);

7) обновить файл секции на диске (¤UPDSEC);

8) обновить файл секции на диске и ждать (¤UPDSECW).

Более того, флаги событий могут использовать несколько процессов, если только эти процессы входят в одну группу. Так, если требуется разработать подсистему, которая требует одновременного выполнения нескольких процессов, можно использовать флаги событий для установления связи между ними и синхронизации их действий.

4.1. Номера флагов событий и кластеры флагов событий

Каждый флаг события имеет уникальный десятичный номер, и все аргументы флагов событий при вызовах программ системного обслуживания ссылаются на данные номера. Например, если при вызове программы системного обслуживания ¤QIO указывается флаг события 1, то после завершения операции

ввода-вывода устанавливается флаг 1.

Чтобы организовать работу с группой флагов событий, данные флаги группируются в кластеры по 32 флага в каждом кластере, и нумеруются справа налево в соответствии с битами с 0 по 31 в длинном слове, кластеры нумеруются с 0 до 3. Диапазон номеров флагов событий включает флаги во всех кластерах: флаг 0 - это первый флаг в кластере 0, флаг 32 - это первый флаг в кластере 1 и т.д.

Имеется два типа кластеров, кластеры общих флагов событий и кластеры локальных флагов событий:

1) кластеры локальных флагов событий можно использовать только внутри одного процесса. Кластеры локальных флагов событий автоматически доступны любому процессу;

2) кластеры общих флагов событий могут разделяться кооперирующимися процессами из одной группы. Перед тем, как процесс может обратиться к кластеру общих флагов событий, он должен быть явным образом "соединен" с этим кластером.

Диапазоны флагов событий и кластеров, к которым они принадлежат, приведены в табл. 26.

Таблица 26

Сводка номеров флагов событий и кластеров

Номер кластера	Номера флагов событий	Описание	Ограничение
0	0 - 31	Кластеры локальных флагов событий процесса для общего использования	Флаги событий с 24 по 31 не используются операционной системой МОС ВП
1	32 - 63		
2	64 - 95	Назначаемый кластер общих флагов событий	Должен быть подсоединён перед использованием
3	96 - 127		

4.1.1. Спецификация номеров флагов событий и кластеров флагов событий

Одни и те же программы системного обслуживания манипулируют флагами как в кластерах локальных флагов событий так и в кластерах общих флагов событий. Поскольку номер флага события включает в себя номер кластера, то при вызове программы системного обслуживания, которая обращается к флагу события, нет необходимости указывать номер кластера.

Если программа системного обслуживания требует в качестве аргумента номер кластера общих флагов событий, то необходимо только задать номер любого флага события, входящего в этот кластер. Так, чтобы прочитать флаги событий из

кластера 1, можно указать любой номер в диапазоне от 32 до 63.

Чтобы предотвратить случайное использование флага события, который уже используется в каком-то месте программы, следует распределять и освобождать локальные флаги событий.

Локальные флаги событий наиболее часто используются совместно с другими программами системного обслуживания. Например, можно использовать программу системного обслуживания установки таймера (@SETIMR) для запроса на установку флага события в конкретное время дня или после истечения определенного интервала времени. Если требуется перевести процесс в состояние ожидания на определенный период времени, то можно указать номер флага события для служебной программы @SETIMR, а затем использовать программу системного обслуживания @WAITFR.

Пример.

```
TIME:   .BLKQ   1           будет содержать интервал времени
                            ожидания

        @SETIMR_S -         установить таймер
                EFN=#33, -
                DAYTIM=TIME
        @WAITFR_S -
                EFN=#33     ждать, пока истечет интервал вре-
                            мени
```

В приведенном примере аргумент DAYTIM задает

64-битовое значение времени.

4.2. Ожидания флагов событий

Три программы системного обслуживания помещают процесс в состояние ожидания до тех пор, пока не будет установлен флаг (или флаги) события:

1) программа системного обслуживания "ждать одного флага события" (¤WAITFR) помещает процесс в состояние ожидания, пока не будет установлен флаг события;

2) программа системного обслуживания "ждать логического "или" флагов событий" (¤WFLOR) помещает процесс в состояние ожидания, пока не будет установлен любой флаг из указанной группы флагов событий;

3) программа системного обслуживания "ждать логического "и" флагов событий" (¤WFLAND) помещает процесс в состояние ожидания, пока не будут установлены все флаги из указанной группы флагов событий.

Другой программой системного обслуживания, которая принимает в качестве аргумента номер флага события, является программа "поставить в очередь запрос на ввод-вывод" (¤QIO). В примере показан программный сегмент, в котором дважды вызывается программа системного обслуживания ¤QIO и используется программа системного обслуживания ¤WFLAND для перевода в состояние ожидания, пока не выполнятся обе операции ввода-вывода, после чего программа продолжает выполнение.

Пример.

```
¤QIO_S  EFN=#1,...  1       выдает первый запрос на
                            ввод-вывод
BSBW    ERROR               контроль на ошибку
¤QIO_S  EFN=#2,...          выдается второй запрос на
                            ввод-вывод
BSBW    ERROR               контроль на ошибку
¤WFLAND_S -   2             ожидать, пока закончатся
                            обе операции
        EFN=#1, -  3
        MASK=#^B0110
BSBW    ERROR               контроль на ошибку
                            продолжить выполнение
```

Примечания:

1. Аргумент флаг события задается для каждого запроса ¤QIO. Оба эти флага события находятся в кластере с номером 0;

2. После того, как оба запроса на ввод-вывод успешно поставлены в очередь, программа вызывает программу системного обслуживания "ждать логического "и" флагов событий" (¤WFLAND) для перевода в состояние ожидания, пока не закончатся обе операции ввода-вывода. В вызове служебной программы аргумент EFN может указывать любой номер флага события в кластере, который должен содержать ожидаемые флаги событий. Аргумент MASK указывает, что ожидаются флаги 1 и 2;

3. Программа системного обслуживания ¤WFLAND (и другие

программы системного обслуживания ожидания) ждет, пока не будет установлен флаг события, а не завершения операции ввода-вывода. Если какое-либо другое событие должно было бы установить требуемые флаги событий, то ожидание флага было бы абсолютно бессмысленным. Использование флагов должно быть четко скоординировано (п. 7.3.1).

4.3. Установка и сброс флагов событий

Программы системного обслуживания, которые используют флаги событий, сбрасывают флаг события, указанный при вызове программы, перед тем, как поставить в очередь запрос на таймер или операцию ввода-вывода. Это гарантирует, что состояние флага события различается процессами. При использовании флагов событий в кластерах локальных флагов событий для других целей, необходимо проследить за начальным состоянием флага.

Программы системного обслуживания "установить флаг события" (¤SETEF) и "сбросить флаг события" (¤CLREF) устанавливают и сбрасывают флаги событий.

Пример.

```
¤CLREF_S        EFN=#32
```

Программы системного обслуживания ¤SETEF и ¤CLREF возвращают коды успешного состояния, которые индицируют, был ли установлен или сброшен указанный флаг при вызове программы. Вызывающая программа, таким образом, может определить, если требуется, предыдущее состояние флага. Возвращаемые коды имеют символические имена SS¤_WASSET и

SS¤_WASCLR.

Флаги событий в кластере общих флагов событий первоначально сброшены в нулевое состояние при создании кластера.

4.4. Кластеры общих флагов событий

Флаги общих событий действуют как линия связи между образами, выполняющимися в различных процессах из одной группы. Флаги общих событий используются в качестве средства синхронизации для других более сложных методов связи, таких как логические имена и глобальные секции.

Прежде, чем какой-либо процесс сможет использовать флаги событий в кластере общих флагов событий, необходимо создать этот кластер. Программа системного обслуживания "присоединить кластер общих флагов событий" (¤ASCEFC) создает кластер общих флагов событий. После того, как кластер создан, другие процессы из той же группы могут вызвать программу ¤ASCEFC, чтобы установить свою связь с этим кластером, после чего они получают доступ к флагам в этом кластере.

Когда создается кластер общих флагов событий, он должен идентифицироваться строкой имени (п. 4.6.1). Каждый процесс, который связывается с кластером, должен использовать то же самое имя для обращения к кластеру. Программа ¤ASCEFC устанавливает соответствие между именем кластера и его номером, который ему назначает процесс.

В примере показано, как процесс может создать кластер общих флагов событий с именем COMMON_CLUSTER и назначить

ему номер 2.

Пример.

```
CLUSTER:
        .ASCID  /COMMON_CLUSTER/    имя кластера

        $ASCEFC_S -                 создать кластер с
                                    с номером 2
                EFN=#65, -
                NAME=CLUSTER
```

После этого другие процессы той же группы могут связаться с этим кластером. Такие процессы при обращении к кластеру должны использовать то же имя в виде строки символов, однако номера кластера, которые они назначают, не обязаны быть теми же самыми.

Кластеры общих флагов событий бывают постоянными или временными. Аргумент PERM программы системного обслуживания $ASCEFC определяет, является ли кластер временным или постоянным.

Временные кластеры требуют, чтобы создающий процесс имел квоту для элементов очереди таймера (квота TQELM). Они удаляются, когда отсоединяются все процессы, связанные с кластером. Отсоединение может быть выполнено либо в явном виде с помощью программы системного обслуживания "отсоединить кластер общих флагов событий" ($DACEFC) либо неявно, когда образ выходит из операционной системы МОС ВП.

Постоянные кластеры требуют, чтобы создающий процесс имел пользовательскую привилегию PRMCEB. Такие кластеры

существуют до тех пор, пока не будут явно помечены для удаления с помощью программы системного обслуживания ¤DLCEFC.

Если все взаимодействующие процессы, которые собираются использовать кластер общих флагов событий имеют необходимую привилегию или квоту для создания кластера, то первый же процесс, вызвавший программу системного обслуживания ¤ASCEFC, создает кластер.

4.5. Отсоединение и удаление кластеров общих флагов событий

Когда процесс больше не нуждается в доступе к кластеру общих флагов событий, он вызывает программу системного обслуживания "отсоединить кластер общих флагов событий" (¤DACEFC). Если все процессы, соединение с временным кластером, вызвали программу ¤DACEFC, то операционная система МОС ВП удаляет этот кластер. Если процесс не отсоединяет себя от кластера явно, то операционная система МОС ВП выполняет неявное отсоединение, когда образ, вызвавший программу ¤ASCEFC, выходит из системы.

Постоянные кластеры, однако, должны явно помечаться для удаления программой системного обслуживания "удалить кластер общих флагов событий" (¤DLCEFC). После того, как кластер помечается для удаления, он не удаляется, пока все соединенные с ним процессы не будут отсоединены.

4.6. Кластеры общих флагов событий в разделяемой памяти

Кластер общих флагов событий в памяти, разделяемой несколькими процессорами, является средством, с помощью которого процессы, выполняющиеся на различных процессорах могут связываться один с другим. Процесс может создать кластер общих флагов событий, используя служебную программу "соединить кластер общих флагов событий (¤ASCEFC), специфицирующую имя кластера, которая размещает кластер в памяти, разделяемой несколькими процессорами (см. п. 4.6.1). Другие процессы на том же или на другом процессоре могут связаться с этим кластером, задавая то же самое имя кластера.

Чтобы создать или удалить кластер общих флагов событий в памяти, разделяемой многими процессорами, требуется пользовательская привилегия SHMEM, однако для подсоединения существующего кластера такой привилегии не требуется.

4.6.1. Имя кластера

Аргумент NAME для программы системного обслуживания "подсоединить кластер общих флагов событий" (¤ASCEFC) идентифицирует кластер, который процесс создает или с которым он связывается. Аргумент NAME задает дескриптор, который указывает на строку символов, определяющую, находится ли кластер в памяти, разделяемой многими процессорами.

Эта строка имеет следующий формат:

[имя-разделяемой-памяти:] имя-кластера

где имя-разделяемой-памяти - идентифицирует память, разделяемую многими процессорами, в которой кластер создается либо уже существует. (Это имя назначается блоку памяти когда он подсоединяется во время генерации системы). Если это поле не включено, то кластер существует или создается в памяти, которая является локальной для того процессора, в котором выполняется вызывающий процесс;

имя-кластера - это имя кластера. Можно выбрать любое допустимое имя длиной от 1 до 15 символов. При этом все процессы, связанные с тем же самым кластером общих флагов событий, должны задавать то же имя.

Можно включать как имя-разделяемой-памяти так и имя-кластера для кластера общих флагов событий в памяти, разделяемой многими процессорами. Однако, если пользователю требуется использовать существующие программы без повторной компиляции или повторной компановки, то он может указать только имя-кластера и оставить операционной системе МОС ВП транслировать его до полной спецификации. Операционная система МОС ВП пытается выполнить трансляцию логического имени строки, заданной аргументом NAME.

Текущая строка с именем просматривается в поисках двоеточия. Если двоеточие найдено внутри текущей строки имени, то принимается, что кластер общих флагов событий должен быть расположен в разделяемой памяти и трансляция

продолжается следующим образом:

1) часть текущей строки имени справа от двоеточия помещается в буфер имя-кластера. Часть текущей строки имени слева от двоеточия становится новой текущей строкой имени;

2) к текущей строке имени добавляется префикс CEF¤ и результат подвергается трансляции логического имени;

3) если результат содержит логическое имя, то шаги 1 и 2 повторяются до тех пор, пока трансляция не кончится успешно или пока число трансляций не превышает число, указанное параметром LNM¤C_MAXDEPTH при генерации операционной системы MOC ВП;

4) префикс CEF¤ удаляется из текущей строки имени, которую уже нельзя транслировать. Это имя становится именем разделяемой памяти. Именем кластера будет являться текущая строка, которая содержится в буфере имя-кластера.

Если кластер общих флагов событий размещается в локальной памяти, то трансляция продолжается следующим образом:

1) к текущей строке имени добавляется префикс CEF¤ и результат подвергается трансляции логического имени;

2) если результат является логическим именем, то повторяется шаг 1 до тех пор, пока трансляция не закончится успешно или количество выполненных трансляций не превысит число, заданное параметром LNM¤C_MAXDEPTH;

3) префикс CEF¤ удаляется из текущей строки имени, которую уже нельзя транслировать. Эта текущая строка имени является именем-кластера.

Например, пользователь сделал следующее назначение логического имени:

Пример 1.

```
¤DEFINE CEF¤CLUS_RT SHRMEM¤1:CLUS_RT
```

Пользовательская программа содержит следующий фрагмент:

Пример 2.

```
NAMEDESC:
        .ASCID  /CLUS_RT/          дескриптор для
                                   логического имени
                                   кластера

        ¤ASCEFC_S -
                ...,NAME=NAMEDESC,...
```

Примечания:

Имеет место следующая трансляция логического имени:

1. К имени CLUS_RT добавляется префикс CEF¤;

2. Имя CEF¤CLUS_RT транслируется в SHRMEM¤1:CLUS_RT. (дальнейшая трансляция логического имени заканчивается по ошибке и строка передается служебной программе).

Имеется два исключения из данного метода трансляции логического имени:

1) если строка имени начинается с символа подчеркивания (_), то операционная система МОС ЭП рассматривает результирующую строку в качестве актуального имени (т.е. не выполняется дальнейшая трансляция);

2) если строка имени является результатом трансляции

логического имени, тогда проверяется, имеет ли эта строка имени "терминальный атрибут". Если строка имени отмечена "терминальным" атрибутом, то операционная система МОС ВП рассматривает результирующую строку в качестве актуального имени (т.е. не выполняется дальнейшей трансляции).

5. Служебные программы AST

Некоторые программы системного обслуживания разрешают процессам требовать их прерывания, если происходит конкретное событие. Поскольку прерывание происходит асинхронно (вне последовательности) по отношению к выполнению процесса, механизм прерывания называется асинхронным системным прерыванием (AST). Прерывание обеспечивает передачу управления пользовательской программе, которая обрабатывает событие.

Следующие программы системного обслуживания обслуживают AST:

1) включить механизм AST (¤SETAST);

2) об'явить AST (¤DCLAST);

3) установить AST восстановления питания (¤SETPRA).

Программы системного обслуживания, которые используют механизм AST, принимают в качестве аргумента адрес подпрограммы обслуживания AST, т.е. подпрограммы, которой должно передаваться управление, когда происходит событие.

Прерывания AST используют следующие программы системного обслуживания:

1) об'явить AST (¤DCLAST);

2) поставить в очередь запрос на блокирование (¤ENQ);

3) получить информацию об устройстве-задании (¤GETDVI);

4) получить информацию о задании-процессе (¤GETJPI);

5) получить информацию о всей системе (¤GETSYI);

6) поставить в очередь запрос на ввод-вывод (¤QIO);

7) установить таймер (¤SETIMR);

8) установить AST восстановления питания (¤SETPRA);

9) обновить файл секции на диске (¤UPDSEC).

Например, если вызывается программа системного обслуживания "установить таймер" (¤SETIMR), то можно указать адрес подпрограммы, которая будет выполняться по истечении интервала времени или в конкретное время дня. Программа системного обслуживания планирует выполнение подпрограммы и возвращает управление, а образ программы продолжает выполнение. Когда происходит затребованное событие, операционная система МОС ВП "доставляет" AST, прерывая процесс и вызывая указанную подпрограмму.

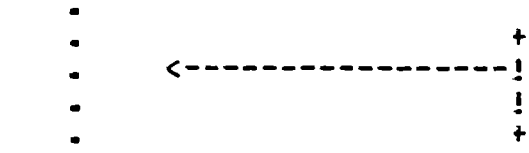
В примере 1 показана типичная программа, которая вызывает программу системного обслуживания ¤SETIMR с запросом прерывания AST, когда произойдет некоторое событие таймера.

Пример 1.

```
NOON:   .BLKQ   1               будет содержать 12:00
                                системного времени
        .ENTRY  LIBRA,0         маска входа для LIBRA



1       ¤SETIMR_S -             установить таймер
                DAYTIM=NOON, -
                ASTADR=TIMEAST
        BSBW    ERROR           контроль на ошибку
```




```
.ENTRY  TIMEAST, ^M<>   ; маска входа для подпрог-
                        раммы AST, обрабатываю-
                        щей запрос таймера
3
        RET             сделано
        .END    LIBRA
```

Примечания:

1. Вызов программы системного обслуживания $SETIMR запрашивает прерывание AST в полдень 12:00.

Аргумент DAYTIM ссылается на квадрослово с именем NOON, которое должно содержать время в системном формате времени (64 бита). Аргумент ASTADR ссылается на TIMEAST, адрес подпрограммы обслуживания AST.

После завершения вызова программы системного обслуживания процесс продолжает выполнение.

2. В 12:00 таймер извещает операционную систему МОС ВП о возникновении заданного события. Операционная система мос вп прерывает выполнение процесса и передает управление подпрограмме обслуживания AST.

3. Пользовательская подпрограмма TIMEAST обрабатывает прерывание. Когда подпрограмма AST заканчивается она выдает инструкцию для возврата управления программе. Программа возобновляет выполнение с той точки, в которой она была прервана.

5.1. Режим доступа для выполнения AST

Каждый запрос на AST связан с режимом доступа программы, из которой запрашивается AST. Таким образом, если образ, выполняющийся в режиме пользователя, запрашивает отметку о событии посредством AST, то подпрограмма обслуживания AST выполняется в пользовательском режиме.

Поскольку прерывания AST, которые запрашиваются пользователем, почти всегда выполняются в режиме пользователя, то ему нет необходимости беспокоиться о режимах доступа. Однако пользователю полезно знать о некоторых подходах операционной системы МОС ВП к доставке AST (подраздел 5.5.).

5.2. Прерывания AST и состояния ожидания процессов

Процесс, который находится в состоянии ожидания, может быть прерван для доставки AST и выполнения подпрограммы обслуживания AST. После того как подпрограмма обслуживания AST закончит выполнение, процесс возвращается в состояние ожидания, если еще действительны те условия, которые вызвали ожидание.

Можно прерывать любые состояния ожидания, кроме приостановленных ожиданий (SUSP) и приостановленных ожиданий обмена (SUSPO).

5.2.1. Ожидание флагов событий

Если процесс ожидает флаг события, то после выполнения подпрограммы обслуживания AST состояние ожидания восстанавливается. Если же по завершении подпрограммы обслуживания AST флаг окажется установленным (например, закончилась операция ввода-вывода), то процесс продолжает выполнение.

5.2.2. Спячка

Процесс может поместить себя в состояние ожидания с помощью программы системного обслуживания "спячка" ($HIBER). Это состояние ожидания может прерваться для доставки AST. После завершения выполнения подпрограммы обслуживания AST процесс продолжает "спячку". Процесс, однако, может "разбудить" себя в подпрограмме обслуживания AST, или быть разбуженным другим процессом, или в результате запланированного по времени запроса на пробуждение. Тогда он продолжает выполнение после завершения подпрограммы обслуживания AST.

Приостановка процесса является другой формой ожидания. Такое ожидание не может быть прервано AST (раздел 8).

5.2.3. Ожидание ресурсов и страничные отказы

Когда процесс выполняет образ, операционная система МОС ЭП может поместить этот процесс в состояние ожидания, пока не станут доступными затребованные ресурсы, или пока не будет считана в память страница из виртуального адресно-

го пространства. Эти ожидания могут прерываться доставкой AST.

5.3. Об'явление прерываний AST

Большинство AST происходят в результате завершения асинхронного события инициализированного программой системного обслуживания (например, запросы ¤QIO или ¤SETIMR), когда процесс запрашивает извещение об этом с помощью AST.

Имеется также программа системного обслуживания, которая создает прерывания AST. Это программа "об'явить AST" (¤DCLAST). С помощью этой программы процесс может об'являть AST только для того же или менее привилегированного режима доступа.

5.4. Подпрограммы обслуживания AST

Подпрограмма обслуживания AST должна быть отдельной программой. Операционная система МОС ВП вызывает AST инструкцией CALLG. Подпрограмма должна возвращать управление, используя инструкцию RET. Если подпрограмма обслуживания модифицирует любые регистры, отличные от R0 или R1, то она должна установить соответствующие биты в маске входа, чтобы содержимое этих регистров было сохранено.

Поскольку нельзя заранее знать, когда подпрограмма обслуживания AST начнет выполняться, программист должен корректно писать эту подпрограмму, чтобы она не модифицировала какие-либо данные или инструкции основной программы

(если только это не является ее функцией).

При входе в подпрограмму обслуживания AST регистр указателя аргумента (AP) указывает на список аргументов, который имеет формат, представленный на рис. 34.

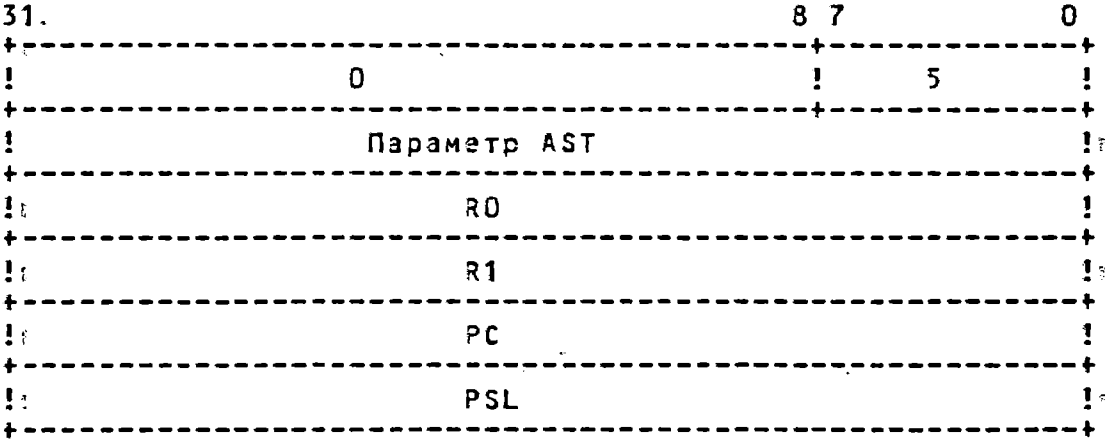


Рис. 34

Регистры R0 и R1, PC и PSL в этом списке совпадают с теми, которые были сохранены в момент прерывания процесса доставкой AST.

Параметр AST является аргументом, который передается подпрограмме обслуживания AST так, чтобы он идентифицировал событие, которое вызывает прерывание AST. При вызове программы системного обслуживания, требующей AST, или при вызове системной служебной программы $DCLAST можно указать это значение для параметра AST. Если это значение не задается, то по умолчанию оно принимается равным 0.

Пример иллюстрирует подпрограмму обслуживания AST. В этом примере прерывания AST ставятся в очередь программой системного обслуживания $DCLAST. Прерывания AST доставляются процессу немедленно, так что подпрограмма обслуживания

вызывается вслед за каждым вызовом программы системного обслуживания ¤DCLAST.

Пример.

```
        .ENTRY  CELESTEF,0          маска входа

1   ¤DCLAST_S -                     AST с параметром = 1
                ASTADR=ASTRTN, -
                ASTPRM=#1

        ¤DCLAST_S -                 AST с параметром = 2
                ASTADR=ASTRTN, -
                ASTPRM=#2

        RET                         вернуть управление
;
ASTRTN: .WORD   0                   маска входа
2       CMPL    #1,4(AP)        проверить, этот параметр AST
        BEQL    10¤             ; если равно 1, то  перейти  к
                                ; 10¤
        CMPL    #2,4(AP)          проверить, этот параметр AST
        BEQL    20¤               если равно 2, то  перейти  к
                                ; 20¤

10¤:.                               обработать первое AST
        RET                         вернуться
20¤                                 обработать второе AST
        RET                         вернуться
```

```
.END CELESTEF
```

Примечание.

1. Программа CELESTEF дважды вызывает программу системного обслуживания =DCLAST, чтобы поставить AST в очередь. Оба AST указывают на подпрограмму обслуживания AST с именем ASTRTN. Однако для каждого вызова передаются различные параметры.

2. Первое действие, которое выполняет подпрограмма AST, состоит в проверке параметра AST, чтобы определить, доставлен он при первом или втором об'явлении. Значение параметра определяет логику выполнения программы. Если имеется несколько различных значений, определяющих несколько различных путей выполнения, то рекомендуется использовать инструкцию CASE языка макроассемблер операционной системы МОС ВП.

5.5. Доставка AST

Возможна такая ситуация, что когда выполнение какого-то условия требует доставить AST, операционная система МОС ВП не может сразу же доставить AST процессу. AST не может быть доставлено процессу, если имеет место одно из следующих условий:

1) в данное время выполняется подпрограмма обслуживания AST с таким же или более привилегированным режимом доступа.

Поскольку во время выполнения подпрограммы обслуживания AST прерывания AST неявно отключены, то одна подпрограмма AST не может быть прервана другой подпрограммой AST, об'явленного для того же режима доступа. Однако она может быть прервана для AST, об'явленного привилегированным режимом доступа;

2) доставка AST для данного режима доступа отключена явно.

Процесс может отключать доставку прерываний AST с помощью программы системного обслуживания "включить механизм AST" ($SETAST). Эта программа системного обслуживания может быть полезной, когда программа выполняет последовательность инструкций, которые не могут быть прерваны для выполнения подпрограммы AST;

3) процесс выполняется с режимом доступа более привилегированным, чем тот с которым об'является AST.

Например, если в результате работы программы системного обслуживания об'явлено AST с режимом пользователя, но в это время программа выполняется с более привилегированным режимом доступа (например, вследствие другого вызова программы системного обслуживания), то AST не доставляется, пока программа снова не будет выполняться в режиме пользователя.

Если при возникновении прерывания AST не может быть доставлено, то оно становится в очередь, пока не исчезнут условия, вызвавшие невозможность доставки. В очереди прерывания AST упорядочиваются по режиму доступа программы, из

которой они были об'явлены. AST, об'явленные из более привилегированных режимов доступа, становятся в начале очереди. Если в очереди находятся более одного AST с совпадающим режимом доступа, то эти AST доставляются в том порядке, в котором они были поставлены в очередь.

6. Программы обслуживания логических имен

Программы обслуживания логических имен операционной системы МОС ЭП обеспечивают технику манипулирования и замещения имен. Логические имена обычно используются для указания устройств или файлов для операций ввода-вывода. Логические имена можно использовать для передачи информации между процессами путем создания логического имени одним процессом в разделяемой таблице логических имен и трансляции этого логического имени в другом процессе. В операционной системе МОС ЭП имеются следующие программы обслуживания логических имен:

1) создать логическое имя (¤CRELNM);

2) создать таблицу логических имен (¤CRELNT);

3) удалить логическое имя (¤DELLNM);

4) транслировать логическое имя (¤TRNLNM).

Операционная система МОС ЭП выполняет специальные программы трансляции логических имен для тех имен, которые связаны со служебными программами ввода-вывода и с программами, которые могут иметь дело со средствами, расположенными в разделяемой (многовходовой) памяти (разделы 4, 7, 11).

6.1. Концепции логических имен

Как следует из названий системных программ обслуживания логических имен, используя эти программы, пользователь может: создавать, удалять и транслировать логические имена, а также создавать и удалять таблицы логических имен. Имеет-

ся несколько концепций, о которых следует знать при использовании системных программ обслуживания логических имен.

5.1.1. Логические имена и эквивалентные имена

Логическое имя - это заданная пользователем строка символов, которая может представлять спецификацию файла, имя устройства, имя таблицы логических имен, специфичную информацию или другое логическое имя. Обычно для целей конкретного процесса пользователь задает логические имена, которые легко использовать и запомнить. Администраторы операционной системы МОС ЭП и привилегированные пользователи выбирают мнемонические имена для файлов, системных устройств и каталогов, к которым часто обращаются все пользователи.

Эквивалентное имя - это строка символов, которая определяет действительную спецификацию файла, имя устройства или строку символов. Эквивалентное имя может также быть логическим именем. В этом случае требуется дальнейшая трансляция, если она допустима, чтобы получить действительное эквивалентное имя.

Многозначное логическое имя, обычно называемое поисковым списком, является именем, которое имеет более одной эквивалентной строки. Каждой эквивалентной строке назначается номер индекса, начиная с нуля.

Логические имена и их эквивалентные строки содержатся в таблице логических имен.

Логические имена могут иметь максимальную длину до 255 символов. Длина эквивалентной строки не может превышать 255 символов. Можно создать пару "логическое имя - эквивалентная строка" следующим образом:

1) либо на командном уровне, используя команды ALLOCATE (распределить), ASSIGN (назначить), DEFINE (определить) или MOUNT (смонтировать) языка DCL;

2) либо в программе, используя программы системного обслуживания "создать логическое имя (¤CRELNM), "создать почтовый ящик и назначить канал" (¤CREMBX) или смонтировать том" (¤MOUNT).

Например, для обращения к терминалу в программе можно было бы использовать символическое имя TERMINAL. Для конкретного прогона программы можно использовать команду DEFINE для задания эквивалентного имени TTA2:.

6.1.2. Таблицы логических имен

Таблицы логических имен содержат пары "логическое имя - эквивалентная строка". К таблице логических имен обращаются по логическому имени.

Таблица логических имен может быть создана либо в адресном пространстве процесса, либо в системном адресном пространстве. Таблица, созданная в адресном пространстве процесса доступна только этому процессу. Таблицы, созданные в системном адресном пространстве, потенциально доступны многим процессам. Некоторые таблицы логических имен содержат заранее определенные логические имена, которые обеспе-

чивают среду для создания, удаления и трансляции логических имен, заданных пользователем. Эти заранее определенные логические имена начинаются с префикса LNM$. Пары "логическое имя - эквивалентное имя" могут обслуживаться в трех типах таблиц логических имен:

1) каталоги таблиц логических имен;

2) таблицы логических имен, принятых по умолчанию;

3) таблицы логических имен, заданных пользователем.

При создании каждого нового процесса для него создаются каталоги таблиц логических имен и таблицы логических имен, принятых по умолчанию.

6.1.2.1. Каталоги таблиц логических имен

Поскольку имена таблиц логических имен являются логическими именами, то имена таблиц тоже должны размещаться в таблицах логических имен. Для этих целей существуют две специальные таблицы, называемые каталогами. Имена таблиц транслируются из этих каталогов таблиц логических имен. Пары "логическое имя - эквивалентное" имя обслуживаются в двух каталогах таблиц:

1) каталог таблиц процесса (LNM$PROCESS_DIRECTORY);

2) каталог таблиц системы (LNM$SYSTEM_DIRECTORY).

Каталог таблиц логических имен процесса содержит имена всех таблиц логических имен, определяемых пользователем для конкретного процесса с помощью программы системного обслуживания $CRELNT. Кроме того, каталог таблиц процесса содержит имена таблиц логических имен, назначенных системой, имя

таблицы логических имен процесса LNM$PROCESS_TABLE и поисковый список логических имен, принятых по умолчанию.

Каталог системных таблиц содержит имена потенциально разделяемых таблиц логических имен и имена таблиц логических имен, назначенных операционной системой МОС ВП. Для создания логического имени в каталоге системных таблиц, требуется привилегия SYSPRV (п. 6.1.3).

В этих каталогах таблиц могут присутствовать логические имена, отличные от имен таблиц логических имен. Максимальная длина логических имен, созданных в любой из этих таблиц не должна превышать 31 символ. Логические имена, созданные в каталогах таблиц, должны состоять из алфавитно-цифровых символов, знаков денежной единицы ($) и подчеркивания (_). Длина эквивалентных строк не должна превышать 255 символов.

6.1.2.2. Таблицы логических имен, принятых по умолчанию

Определенные таблицы логических имен создаются для процесса или назначаются ему при создании процесса. Эти таблицы называются таблицами логических имен, принятых по умолчанию. Вновь созданный процесс снабжается этими таблицами по умолчанию. В таких таблицах обслуживаются пары "логическое имя - эквивалентное имя".

Каждая таблица логических имен, принятых по умолчанию, имеет связанное с ней логическое имя. Чтобы поместить логическое имя в таблицу логических имен, необходимо указать

имя таблицы логических имен. В табл. 27 перечислены имена таблиц логических имен, принятых по умолчанию, и логические имена используемые для обращения к ним.

Таблица 27

Таблица	Имя	Логическое имя
Процесс	LNM¤PROCESS_TABLE	LNM¤PROCESS
Задание	LNM¤JOB_XXXXXXXX	LNM¤JOB
Группа	LNM¤GROUP_GGGGGG	LNM¤GROUP
Система	LNM¤SYSTEM_TABLE	LNM¤SYSTEM

Буква x представляет шестнадцатеричное восьмизначное число, которое уникально идентифицирует таблицу логических имен задания.

Буква G представляет восьмеричное шестизначное число, которое содержит номер группы пользователей.

Максимальная длина логических имен, созданных в этих таблицах, не должна превышать 255 символов без ограничения на типы используемых символов. Длина эквивалентных строк не должна превышать 255 символов.

Таблица логических имен процесса LNM¤PROCESS_TABLE содержит имена, используемые исключительно процессом. Таблица логических имен процесса создается и существует для каждого процесса в операционной системе МОС ВП. Некоторые элементы таблицы логических имен процесса создаются системными программами, выполняющимися в более привилегированных режимах доступа. Эти элементы квалифицируются режимом дос-

тупа, из которого был создан данный элемент. Таблица логических имен процесса содержит постоянные логические имена, связанные с процессом (табл. 28).

Таблица 28

Логическое имя	Значение
SYS$INPUT	Входной поток, по умолчанию
SYS$OUTPUT	Выходной поток, по умолчанию
SYS$COMMAND	Первоначальный входной поток первого уровня (SYS$INPUT)
SYS$ERROR	Устройство, принимаемое по умолчанию, на которое операционная система МОС ВП пишет сообщения об ошибках

Логическое имя SYS$COMMAND создается только для процесса, выполняющего программу LOGINOUT.

Большинство элементов в таблице логических имен процесса создаются в режимах пользователя и супервизора. В примере 1 показано, как в режиме пользователя образ создает принадлежащее процессу логическое имя ABC, которое представляет эквивалентные строки XYZ и DEF. Всякий раз, когда вызывается элементный код LNM$_STRING аргумента ITMLST следующей эквивалентной строке назначается значение индекса. Вновь созданное логическое имя и его эквивалентная строка содержатся в таблице логических имен процесса LNM$PROCESS_TABLE.

Пример 1.

```
LOGDESC:
        .ASCID                  /ABC/
EQVNAM1:
        .ASCII                  /XYZ/
EQVLEN1=
        .-EQVNAM1
EQVNAM2:
        .ASCII                  /DEF/
EQVLEN2=
        .-EQVNAM2
TABDESC:
        .ASCID                  /LNM$PROCESS/
CRELST:
        .WORD                   EQVLEN1
        .WORD                   LNM$_STRING
        .ADDRESS                EQVNAM1
        .LONG                   0
        .WORD                   EQVNAM2
        .WORD                   LNM$_STRING
        .ADDRESS                EQVNAM2
        .LONG                   0
        .LONG                   0
$CRELNM_S-
        LOGNAM = LOGDESC,-        логическое имя
        TABNAM = TABDESC,-        таблица
        ITMLST = CRELST           эквивалентные строки
```

В примере 2 показано создание на языке DCL логического имени в режиме супервизора.

Пример 2.

```
¤ DEFINE/SUPERVISOR_MODE/TABLE=LNM¤PROCESS ABC XYZ, DEF
```

Логические имена процесса, созданные в режиме пользователя, удаляются, когда процесс выполняет прогон образа. Это поведение иллюстрируется в примере 3.

Пример 3.

```
¤ DEFINE/USER ABC XYZ
¤ SHOW TRANSLATION ABC
    ABC     =     XYZ
¤ DIRECTORY
¤ SHOW LOGICAL ABC
    ABC     =     (неопределено)
```

Команда DIRECTORY языка DCL выполняет прогон образа. В это время удаляются все логические имена, принадлежащие процессу и созданные в режиме пользователя, включая логическое имя ABC.

Таблица логических имен задания является разделяемой таблицей, доступной всем процессам в пределах того же дерева заданий. Когда создается отсоединенный процесс, для этого процесса и всех его потенциальных подпроцессов создается таблица логических имен задания. В это же время в каталоге таблиц логических имен процесса LNM¤PROCESS_DIRECTORY создается логическое имя LNM¤JOB, принадлежащее процессу. Логическое имя LNM¤JOB транслируется в имя таблицы логических имен задания.

Когда создается подпроцесс, создается только принадлежащее процессу логическое имя LNM$JOB, поскольку для основного процесса уже существует таблица логических имен задания.

Таблица логических имен задания содержит три логических имени, постоянно принадлежащие процессам, которые выполняют программу LOGINOUT (табл. 29).

Таблица 29

Логические имена	Значение
SYS$LOGIN	Первоначальное устройство, принятое по умолчанию и каталог
SYS$LOGIN_DEVICE	Первоначальное устройство, принятое по умолчанию
SYS$SCRATCH	Устройство и каталог, принятые по умолчанию, на которые записываются временные файлы

Таким образом, вместо создания этих логических имен в таблице логических имен процесса LNM$PROCESS_TABLE для каждого процесса в пределах дерева заданий, программа LOGINOUT создает эти логические имена один раз, когда эта программа выполняется для процесса в корне дерева заданий.

Таблица логических имен задания может содержать следующие необязательные логические имена:

1) логическое имя, связанное с вновь созданным времен-

ным почтовым ящиком;

2) логическое имя, связанное с томом, который монтируется как личный.

Таблица логических имен группы содержит имена, которые могут использовать взаимосвязанные процессы одной и той же группы. Для добавления имени в таблицу логических имен группы требуется привилегия SYSPRV (см. п. 6.1.3.).

Таблица логических имен группы создается, когда она требуется. Однако в каталоге процесса LNM¤PROCESS_DIRECTORY, для каждого процесса существует логическое имя LNM¤GROUP. Это логическое имя транслируется в имя таблицы логических имен группы.

Таблица логических имен системы, содержит имена, к которым могут иметь доступ все процессы операционной системы МОС ВП . Эта таблица включает все имена, принимаемые по умолчанию, для всех логических имен, назначенных операционной системой МОС ВП. Для добавления и удаления логических имен из таблицы системы требуются привилегии SYSNAM и SYSPRV (см. п. 6.1.3).

6.1.2.3. Таблицы логических имен, определенные пользователем

Таблицы, принадлежащие процессу, а также разделяемые таблицы можно создавать, вызывая из программы пользователя программу системного обслуживания ¤CRELNT. Однако для создания разделяемой таблицы требуется привилегия SYSPRV (см.

п. 6.1.3).

Логические имена из таблицы, принадлежащей процессу, может использовать только процесс, создавший эту таблицу.

Таблицы логических имен создаются либо с помощью программы системного обслуживания ¤CRELNT, либо с помощью команды CREATE/NAME_TABLE языка DCL. Разделяемой таблицей могут пользоваться процессы, отличные от процесса, создавшего эту таблицу, если им разрешен доступ.

Максимальная длина логических имен, создаваемых в пользовательских таблицах логических имен, не должна превышать 255 символов. Такое же ограничение наложено на длину эквивалентных строк.

6.1.3. Привилегии

Определенные функции системных программ, обслуживающих логические имена, ограничены для пользователей специальными привилегиями. Операционная система МОС ВП проверяет в файле авторизации пользователей (UAF) привилегии конкретного пользователя, которые приписываются ему администратором системы. Операционная система МОС ВП также проверяет право доступа на чтение, запись и удаление. Привилегии позволяют пользователям выполнять функции, приведенные в табл. 30.

Таблица 30

Сводка привилегий

Привилегия	Функция
GRPNAM	Создание или удаление логического имени в таблице логических имен своей группы
GRPPRV	Создание или удаление логического имени в таблице логических имен своей группы
SYSNAM	Создание логических имен режима управления или ядра. Создание или удаление логических имен в своей таблице логических имен системы. Удаление логического имени или таблицы в более привилегированном режиме доступа
SYSPRV	Создание или удаление логического имени в таблице логических имен своей группы. создание или удаление логического имени в таблице логических имен системы. создание разделяемой таблицы

6.1.4. В основном атрибуты, задаваемые с помощью системных программ, обслуживающих логические имена, выполняют две функции:

1) влияют на создание логических имен или управляют работой программ системного обслуживания;

2) влияют на трансляцию логических имен и эквивалентных строк.

Атрибуты, которые влияют на создание логических имен указываются в необязательном аргументе ATTR при вызове программы системного обслуживания:

LNM¤M_CONFINE - препятствует копированию логического имени из таблицы логических имен процесса в порожденный подпроцесс. Подпроцесс может быть создан командой SPAWN языка DCL или программой LIB¤SPAWN исполнительной библиотеки. Этот атрибут может быть определен только в программе системного обслуживания ¤CRELNM или ¤CRELNT;

LNM¤M_NO_ALIAS - препятствует созданию дубликата логического имени в указанной таблице логических имен во внешнем режиме доступа. Если другое логическое имя уже существует в таблице для внешнего режима доступа, то оно удаляется.

Если этот атрибут задается при вызове программы системного обслуживания ¤CRELNT, то он препятствует созданию таблицы логических имен для внешнего режима доступа в каталоге таблиц, если такое имя таблицы уже существует в каталоге.

Этот атрибут указывается только при вызове программ системного обслуживания ¤CRELNM или ¤CRELNT;

LNM¤M_CREATE_IF - препятствует созданию таблицы логических имен, если указанная таблица уже существует для заданного режима доступа в соответствующем каталоге таблиц.

Этот атрибут указывается только при вызове программы системного обслуживания ¤CRELNT;

LNM¤M_CASE_BLIND - управляет процессом трансляции и заставляет программу ¤TRNLNM игнорировать разницу между большими или строчными буквами при поиске логического имени. Этот аргумент указывается только при вызове программы системного обслуживания ¤TRNLNM.

При вызове программы системного обслуживания ¤CRELNM в аргументе ITMLST с помощью элементного кода LNM¤_ATTRIBUTES можно задать необязательные атрибуты трансляции LNM¤M_CONCEALED и LNM¤M_TERMINAL, связанные с логическими именами и эквивалентными строками. Если элементный код LNM¤_ATTRIBUTES задается через программу ¤TRNLNM, то операционная система МОС ЭП возвращает текущие атрибуты, связанные с логическим именем и эквивалентной строкой, в текущем значении индекса:

LNM¤M_CONCEALED - указывает, что эквивалентная строка с текущим значением индекса для данного логического имени является именем устройства, понятым в системе управления данными;

LNM¤M_CONFILE - указывает, что порожденный процесс не может использовать данное логическое имя. Подпроцессы создаются командой SPAWN языка DCL или библиотечной программой LIB¤SPAWN;

LNM¤M_CRELOG - указывает, что логическое имя было создано с помощью программы системного обслуживания ¤CRELOG;

LNM$M_EXISTS - указывает, что для указанного значения индекса существует эквивалентная строка;

LNM$M_NO_ALIAS - указывает, что, если логическое имя уже существует в таблице, то оно не может быть создано в этой таблице для внешнего режима доступа;

LNM$M_TABLE - указывает, что логическое имя является именем таблицы логических имен;

LNM$M_TERMINAL - указывает, что эквивалентные строки больше нельзя транслировать.

Атрибуты множественных эквивалентных строк не должны быть одинаковыми.

6.1.5. Квоты таблиц логических имен

Квота таблицы логических имен - это число байтов, выделенных в памяти для логических имен, содержащихся в этой таблице. Квоты таблиц логических имен устанавливаются на следующих этапах:

1) инициализация операционной системы МОС ВП;

2) создание процесса;

3) создание таблицы логических имен.

Каждая таблица логических имен имеет связанную с ней квоту, которая ограничивает число байтов памяти (либо из пула процесса, либо из страничного пула системы), которые выделяются для логических имен, определяемых в таблице. Квота таблицы устанавливается при создании таблицы.

Если не указана никакая квота, то вновь созданная таблица имеет неограниченную квоту. Эта таблица может расши-

ряться и при этом потреблять всю память доступную процессу или операционной системе МОС ВП, а все пользователи, имеющие доступ на запись к такой разделяемой таблице, могут вызвать неограниченное потребление системного страничного пула.

6.1.5.1. Квоты каталогов таблиц

При инициализации операционной системы МОС ВП для системного каталога таблиц LNM$SYSTEM_DIRECTORY автоматически устанавливается неограниченная квота.

При подключении пользователя к операционной системе МОС ВП для каталога таблиц процесса LNM$PROCESS_DIRECTORY автоматически устанавливается неограниченная квота.

6.1.5.2. Квоты таблиц логических имен, принимаемых по умолчанию

Таблицы логических имен процесса, группы и системы имеют неограниченные квоты.

Поскольку таблица логических имен задания является разделяемой таблицей, и для создания в ней логических имен не требуется никаких специальных привилегий, квота, выделяемая этой таблице логических имен, утверждается при ее создании. Для указания квоты таблицы логических имен задания во время ее создания существуют три механизма:

1) для всех процессов, которые активизируют программу LOGINOUT, квота для логических имен задания берется из фай-

ла авторизации. Можно изменять квоту для таблицы логических имен задания используя команду AUTORIZE/JTQUOTA=;

2) для всех процессов, которые активизируют программу LOGINOUT, квота для таблицы логических имен задания может быть задана как элемент списка квот PQL¤_JTQUOTA при вызове программы системного обслуживания "создать процесс" (¤CREPRC). Если отсоединенный процесс необходимо создать с помощью команды RUN/DETACHED языка DCL, то для указания элемента списка квот программы ¤CREPRC используется команда RUN/JOB_TABLE_QUOTA;

3) для всех процессов, которые не активизируют программу LOGINOUT и не указывают элемент списка квот PQL¤_JTQUOTA при вызове программы ¤CREPRC, квота для таблицы логических имен задания берется из динамического параметра генерации системы (SYSGEN) PQL¤DJTQUOTA. Обслуживающую программу SYSGEN можно использовать для отображения значений параметров PQL¤_DJTQUOTA и PQL¤_MJTQUOTA, которые задают квоту, принимаемую по умолчанию, и минимальную квоту, соответственно.

6.1.5.3. Квоты таблиц логических имен, заданных пользователем

Таблицу логических имен, заданных пользователем, можно создавать либо с явно ограниченной квотой, либо без ограничений квоты.

Наличие квот таблиц логических имен, определенных пользователем, устраняет необходимость в привилегиях, нап-

ример, SYSNAM или GRPNAM, для управления использованием страничного пула при создании логических имен в разделяемой таблице.

6.1.6. Соглашения о форматах логических имен и эквивалентных имен

При создании и трансляции логических имен и эквивалентных имен операционная система МОС ВП использует специальные соглашения.

Если строке логического имени, которая присутствует в служебных программах ввода-вывода, предшествует символ подчеркивания (_), то программы ввода-вывода не выполняют трансляцию логического имени, следующего за символом подчеркивания, считая это логическое имя именем физического устройства.

При подключении пользователя операционная система мос вп создает элементы в таблице логических имен по умолчанию для постоянных файлов процесса. Эквивалентным именам для этих элементов (например, SYS$INPUT и SYS$OUTPUT) предшествует четырехбайтовый заголовок, который содержит следующее:

1) байт 0 ^X1B знак переключения;
2) байт 1 ^X00;
3) байты 2-3 внутренний идентификатор файла (IFI) системы управления данными СУД-32.

За этим заголовком следует строка эквивалентного имени. Если любая из пользовательских программ должна трансли-

ровать логические имена, назначенные операционной системой MOC ВП, то программа должна быть готовой проверить наличие этого заголовка и затем использовать только нужную часть эквивалентной строки. В примере фрагмент программы демонстрирует этот метод.

Пример.

```
ILST:
        .WORD           LNM$C_NAMLENGTH
        .WORD           LNM$_STRING
        .LONG           RESSTRING
        .LONG           RESDESC
        .LONG           0
TABDESC:
        .ASCID          /LNM$FILE_DEV/      имя таблицы
                                            файлов/устройств
LOGDESC:                                    логическое имя,
                                            которое необходи-
                                            мо транслировать
        .ASCID          /INPUT_DEVICE/
RESDESC:                                    дескриптор для
                                            результирующей
                                            строки
        .LONG           LNM$C_NAMLENGTH     размер результи-
                                            рующей строки
        .ADDRESS -                          адрес результи-
                        RESSTRING           рующей строки
RESSTRING:.                                 назначение резу-
```

```
                                                  льтирующей строки
        .BLKB           LNM$C_NAMLENGTH

        $TRNLNM_S                                 транслировать ло-
                        LOGNAM=LOGDESC,-;         гическое имя
                        TABNAM=TABDESC,-
                        ITMLST=ILST
        BLBC            R0,ERR                    перейти по ошибке
        CMPW            RESSTRING, ^X001B           является ли пер
                                                  вый символ знаком
                                                  переключения?
        BNEQ            1$                        если нет, то пе-
                                                  рейти к 1$
        SUBW            #4,RESDESC                если да, то вы-
                                                  честь 4 из длины
        ADDL            #4,RESDESC+4              и добавить 4 к ад
                                                  ресу строки
1$:
```

6.1.7. Спецификация поискового списка таблиц логических имен

Логические имена существуют как элементы в таблицах логических имен. Когда необходимо создать, удалить или транслировать логическое имя, должно быть представлено имя, которое обозначает таблицу, содержащую логическое имя. Данное имя обладает одной из следующих характеристик:

1) это имя таблицы логических имен;

2) это имя таблицы логических имен, которое итеративно транслируется в процессе или каталоге системных таблиц в имя таблицы логических имен;

3) это многозначное логическое имя, которое итеративно транслируется в имена различных таблиц логических имен. Многозначное логическое имя также называют поисковым списком. Таблицы используются в том порядке, в котором они перечислены в списке.

Для некоторых таблиц логических имен существуют заранее определенные логические имена. Эти заранее определенные имена начинаются с перфикса LNM$. Пользователь может переопределить эти имена, чтобы модифицировать порядок поиска или используемые таблицы.

Вместо фиксированного набора таблиц логических имен и жестко определенного порядка (процесс, задание, группа, система) поиска в этих таблицах пользователь может указать, какие таблицы следует просматривать и в каком порядке. Для указания порядка просмотра используются логические имена в каталоге таблиц. По соглашению, каждый класс использования логических имен, например, спецификация файла или устройства, пользуется для этой цели конкретным заранее определен-

ным именем.

Например, LNM$FILE_DEV является именем таблицы логических имен, которое используется при трансляции имен спецификаций файлов или устройств с помощью системы управления данными или программ обслуживания ввода-вывода. Это имя должно транслироваться в списке из одного или более имен таблиц логических имен. Полученный список задает те таблицы, которые должны просматриваться при трансляции спецификаций файлов.

По умолчанию LNM$FILE_DEV указывает, что должны просматриваться все таблицы процесса, задания, группы и системы в порядке этого перечисления, и возвращается первое найденное соответствие.

Имена таблиц логических имен транслируются из двух таблиц: каталога таблиц логических имен процесса LNM$PROCESS_DIRECTORY и каталога таблиц логических имен системы LNM$SYSTEM_DIRECTORY. В одной из этих таблиц должно быть определено имя LNM$FILE_DEV.

Таким образом, если в таблицах процесса и группы существуют идентичные логические имена, то, поскольку элемент таблицы процесса находится первым, таблицы задания и группы не просматриваются. Когда осуществляется поиск в таблице логических имен процесса, элементы просматриваются в порядке методов доступа, вначале режим пользователя, затем режим супервизора и т.д.

Если необходимо изменить список таблиц, используемых спецификаций файлов и устройств, то можно переопределить

LNM¤FILE_DEV в каталоге процесса LNM¤PROCESS_DIRECTORY.

6.2. Создание логических имен - ¤CRELNM

Чтобы создать логическое имя в программе, программист должен обеспечить дескрипторы символьных строк для строк имен, выбрать таблицу, которая будет содержать логическое имя и использовать служебную программу ¤CRELNM, как показано в примере.

Пример.

```
LOGDESC:
        .ASCID                  /TTY/
TABDESC:
        .ASCID                  /LNM¤JOB/
LNMATTR:
        .LONG                   LNM¤_TERMINAL
CRELST:
        .WORD                   4
        .WORD                   LNM¤_ATTRIBUTES
        .ADDRESS                LNMATTR
        .LONG                   0
        .WORD                   EQVLEN
        .WORD                   LNM¤_STRING
        .ADDRESS                EQVNAM
        .LONG                   0
        .LONG                   0
EQVNAM:
        .ASCII                  /TTA2:/
```

```
EQVLEN=
        .-EQVNAM



$CRELNM_S -
        LOGNAM = LOGDESC,-
        TABNAM = TABDESC,-
        ATTR   = TABDESC,-
        ITMLST = CLELST
```

В примере атрибут трансляции задан как терминальный. Этот атрибут указывает, что итеративная трансляция логического имени TTY заканчивается, когда возвращается эквивалентная строка TTA2. Кроме того, поскольку не задан аргумент ACMODE, то режим доступа логического имени TTY совпадает с режимом доступа вызывающего образа.

6.2.1. Дублирование логических имен

Таблица логических имен может содержать элементы для одного логического имени в разных режимах доступа. Различные таблицы логических имен могут содержать элементы одного и того же логического имени.

Во всех остальных случаях для конкретного логического имени может быть только один элемент в таблице логических имен.

Любое число логических имен может иметь одинаковые эквивалентные имена.

На рис. 35-37 логическое имя TERMINAL транслируется по-разному, в зависимости от того, в какой таблице логических имен оно специфицировано.

```
-------------------------------------------------------------
Логическое имя      Эквивалетное имя        Режим доступа
-------------------------------------------------------------
INFINE    --->  DM1:[HIGGINS]TEST.DAT     Режим супервизора
OUTFILE   --->  DM1:[HIGGINS]TEST.OUT     Режим супервизора
TERMINAL  --->  TTA2:                     Режим супервизора
   .                                            .
   .
   .
```

Рис. 35

Таблица логических имен процесса приравнивает логическое имя TERMINAL конкретному терминалу TTA2. Логические имена INFILE и OUTFILE приравниваются спецификациям диска. Эти логические имена были созданы в режиме супервизора.

Чтобы определить эквивалентную строку для логического имени TERMINAL необходимо выдать следующую команду:

```
$ SHOW LOGICAL TERMINAL
```

операционная система МОС ВП вернет эквивалентную строку:

```
TTA2:
```

```
-------------------------------------------------------------
Логическое имя     Эквивалентное имя       Режим доступа
-------------------------------------------------------------
SYS$LOGIN      ---> DBA9:[HIGGINS]      Режим управления
TERMINAL       ---> VTA14:              Режим пользователя
   .                    .                      .
   .                                           .
   .
```

Рис. 36

Таблица логических имен задания определяет логический

терминал TERMINAL как виртуальный терминал VTA14. Логическое имя SYS$LOGIN является устройством и каталогом для процесса при подключении к операционной системе МОС ВП. Логическое имя SYS$LOGIN определяется в режиме управления.

Чтобы определить эквивалентную строку для логического имени TERMINAL, заданного на рис. 36, необходимо выдать следующую команду:

```
¤ SHOW LOGICAL/JOB TERMINAL
```

Операционная система МОС ВП возвращает эквивалентную строку: VTA14:, полученную в результате трансляции.

```
-------------------------------------------------------------
Логическое имя     Эквивалентное имя      Режим доступа
-------------------------------------------------------------
TERMINAL      ---> MBA407:                 Режим супервизора
XYZ           ---> DISK1:                  Режим супервизора
              ---> DISK3:                  Режим супервизора
   .
   .
   .
```

Рис. 37

Таблица логических имен LOG_TBL, заданная пользователем, содержит опрделение логического имени TERMINAL как устройства почтового ящика MBA407:. Многозначное логическое имя XYZ имеет две трансляции - DISK1 и DISK3.

Чтобы определить эквивалентную строку для логического имени TERMINAL, показанного на рис. 37, необходимо выдать следующую команду:

```
¤ SHOW LOGICAL/TABLE=LOG_TBL TERMINAL
```

Операционная система МОС ВП возвращает эквивалентное имя: MBA407:. Чтобы использовать данное определение логи-

ческого имени TERMINAL в качестве спецификации файла или устройства, необходимо переопределить имя таблицы LNM$FILE_DEV так, чтобы обращаться к таблице логических имен, определенной пользователем, следующим образом:

$ DEFINE/TABLE=LNM$PROCESS_DIRECTORY LNM$FILE_DEV -
LOG_TBL, LNM$PROCESS, LNM$JOB, LNM$SYSTEM

Команда DEFINE языка DCL используется для переопределения поискового списка по умолчанию LNM$FILE_DEV.

Квалификатор /TABLE задает таблицу LNM$PROCESS_DIRECTORY, которая будет содержать переопределенный поисковый список. Операционная система МОС ВП будет выполнять поиск в таблицах, определенных списком LNM$FILE_DEV, в следующем порядке: LOG_TBL, LNM$PROCESS_TABLE, LNM$JOB и LNM$SYSTEM_TABLE.

Логическое имя		Эквивалентное имя
SYS$LIBRARY	--->	SYS$SYSROOT:[SYSLIB]
SYS$SYSTEM	--->	SYS$SYSROOT:[SYSEXE]
.		
.		
.		

Рис. 38

Системная таблица логических имен содержит назначенные операционной системой МОС ВП логические имена, доступные всем процессам в системе. Например, к логическим именам SYS$LIBRARY и SYS$SYSTEM, показанным на рис.38, может получить доступ любой пользователь, для того чтобы использовать устройство и каталог содержащие системные файлы.

6.2.1.1. Замена логических имен

Если логическому имени TERMINAL соответствует эквивалентное имя TTA2, в таблице процесса, а затем процесс приравнивает логическое имя TERMINAL к TTA3, то эквивалентное имя - TTA2 заменяется новым эквивалентным именем. Код успешного состояния возврата SS¤_SUPERSEDE указывает, что новый элемент заменил старый.

Определения логического имени TERMINAL в таблице задания и таблице LOG_TBL, определенной пользователем, не меняются.

6.3. Создание таблиц логических имен - ¤CRELNT

Программа системного обслуживания ¤CRELNT создает таблицы логических имен. Таблицы логических имен могут создаваться в любом режиме доступа в зависимости от привилегий вызывающего процесса. Задаваемое пользователем логическое имя, идентифицирующее вновь создаваемую таблицу логических имен запоминается в каталоге таблиц процесса LNM¤PROCESS_DIRECTORY.

6.3.1. Разделяемые таблицы логических имен

Если пользователь имеет привилегию SYSPRV, то он может создавать разделяемые таблицы логических имен. Он может назначить защиту для этих таблиц с помощью, аргумента PROMSK программы системного обслуживания ¤CRELNT. Аргумент

PROMSK разрешает специфицировать тип доступа для системы, владельца, группы и всех пользователей следующим образом:

1) привилегия на доступ с чтением разрешает доступ к именам в таблице логических имен;

2) привилегия на доступ с записью разрешает создание и удаление имен из таблицы логических имен;

3) привилегия на доступ с удалением разрешает удаление таблицы логических имен.

Примечание. Бит защиты "E" не используется.

Если аргумент опущен, то системе и владельцу гарантирован полный доступ, а группе и всем пользователям все виды доступа запрещены.

6.3.2. Вызов программы системного обслуживания =CRELNT

Пример иллюстрирует вызов программы системного обслуживания.

Пример.

```
TABDESC:
        .ASCID          /LOG_TABLE/
PARDESC:
        .ASCID          /LNM=PROCESS_DIRECTORY/
TAB_ATTR:
        .LONG           LNM=M_CONFINE
TAB_QUOTA:
        .LONG           5000
```

¤CRELNT¤S -

```
        TABNAM = TABDESC,-          имя таблицы
        PARTAB = PARDESC,-          порождающая таблица
        ATTR   = TAB_ATTR,-         атрибуты
        QUOTA  = TAB_QUOTA          квота
```

В примере создается определенная пользователем таблица LOG_TABLE с явно заданной квотой в 5000 байтов. Имя вновь созданной таблицы является элементом в таблице LNM¤PROCESS_DIRECTORY. Поскольку атрибут CONFINE связан с таблицей логических имен, эту таблицу нельзя копировать из процесса в порожденные им подпроцессы.

6.4. Удаление логических имен - ¤DELLNM

Программа системного обслуживания ¤DELLNM удаляет элементы из таблицы логических имен. При написании в программе вызова программы системного обслуживания ¤DELLNM, можно указать одно логическое имя для удаления, либо указать, что требуется удалить все логические имена из конкретной таблицы. Следующий пример иллюстрирует вызов, который удаляет логическое имя TERMINAL из таблицы логических имен задания.

Пример.

```
LOGDESC:
        .ASCID          /TERMINAL/
TABDESC:
        .ASCID          /LNM¤JOB/
```

$DELLNM_S -

LOGNAM = LOGDESC, -

TABNAM = TABDESC, -

Имена таблиц логических имен или логические имена, размещенные в таблице логических имен, принадлежащей процессу с помощью команд языка DCL ASSIGN/USER и DEFINE/USER языка DCL, автоматически удаляются при завершении работы образа. Элементы, сделанные из потока команд помещаются в таблицу интерпретатором команд. Это элементы режима супервизора и они не удаляются при завершении работы образа (кроме логических имен, определенных командами ASSIGN/USER и DEFINE/USER языка DCL).

6.5. Трансляция логических имен - $TRNLNM

Программа системного обслуживания $TRNLNM транслирует логическое имя в эквивалентную строку. Кроме того, программа $TRNLNM возвращает информацию о логическом имени и эквивалентной строке.

Таблицы, в которых осуществляется поиск логического имени, указываются в аргументе при вызове программы системного обслуживания $TRNLNM.

Этот аргумент может быть либо именем таблицы логических имен, либо логическим именем, которое транслируется в список из одной или более таблиц логических имен.

Поскольку логические имена могут иметь много эквивалентных строк, то можно задавать, какую именно эквивалентную строку желательно получить.

Ряд программ системного обслуживания, которым требуется имя устройства, принимают логическое имя и транслируют его итеративно, пока не будет получено имя физического устройства (или пока не будет выполнено то число трансляций логического имени, которое принято в операционной системе MDC ВП по умолчанию). Эти системные программы неявно задают имя таблицы логических имен LNM¤FINE_DEV (п. 6.1.7). Следующие программы системного обслуживания автоматически выполняют итеративную трансляцию логического имени:

1) ¤ALLOC - распределить устройство;

2) ¤ASSIGN - назначить канал ввода-вывода;

3) ¤BRDCST - разослать сообщение;

4) ¤CREMBX - создать почтовый ящик;

5) ¤DALLOC - освободить устройство;

6) ¤DISMOU - демонтировать том;

7) ¤GETDVI - получить информацию об устройстве/томе;

8) ¤MOUNT - смонтировать том.

Однако во многих случаях программа должна выполнить трансляцию логического имени, чтобы получить для него эквивалентное имя. В таком случае необходимо предоставить имя таблицы или таблиц, в которых требуется выполнять поиск. Программа системного обслуживания ¤TRNLNM ("транслировать логическое имя") просматривает заданные пользователем таблицы логических имен в поисках указанного логического имени и возвращает эквивалентное имя. Кроме того, программа ¤TRNLNM возвращает атрибуты (которые могут и не указываться) для логического имени и эквивалентной строки.

В примере 1 показан вызов программы системного обслуживания $TRNLNM для трансляции логического имени ABC.

Пример 1.

```
LOGDESC:
        .ASCID          /ABC/
TABDESC:
        .ASCID          /LMN$FILE_DEV/
EQVBUF1:
        .BLKB           LMN$C_NAMLENGTH
EQVDESC1:
        .LONG           0
        .ADDRESS        EQVBUF1
EQVBUF2:
        .BLKB           LMN$C_NAMLENGTH
EQVDESC2:
        .LONG           0
        .ADDRESS        EQVBUF2
TRNLIST:
        .WORD           LMN$C_NAMLENGTH
        .WORD           LNM$_STRING
        .ADDRESS        EQVBUF1
        .ADDRESS        EQVDESC1
        .WORD           LMN$C_NAMLENGTH
        .WORD           LNM$_STRING
        .ADDRESS        EQVBUF2
        .ADDRESS        EQVDESC2
        .LONG           0
```

```
TRNATTR:
        .LONG           LNM$M_CASE_BLIND
    .


$TRNLNM_S       -
        LOGNAM  =  LOGDESC,-
        TABNAM  =  TABDESC,-
        ATTR    =  TRNATTR,-
        ITMLST  =  TRNLIST
```

Вызов программы системного обслуживания $TRNLNM приводит к трансляции логического имени ABC. Кроме того, аргумент TABNAM задает LNM$FILE_DEV в качестве поискового списка, который будет использовать программа $TRNLNM для поиска логического имени ABC. Логическому имени ABC назначены две эквивалентные строки. Элементный код LNM$_STRING в аргументе ITMLST указывает программе $TRNLNM искать эквивалентную строку с текущим значением индекса. Элементный код LNM$_STRING вызывается дважды. Эквивалентные строки помещаются в два выходных буфера EQVBUF1 и EQVBUF2, описанные с помощью ссылки TRNLIST.

Атрибут LNM$M_CASE_BLIND управляет процессом трансляции. Программа $TRNLNM ищет эквивалентные строки, не обращая внимание на разницу в строчных и заглавных буквах.

Длина выходной строки эквивалентного имени записывается в первое слово дескриптора строки символов. Этот дескриптор может быть затем использован в качестве входного для

другой программы системного обслуживания.

7. Программы обслуживания ввода-вывода

Для выполнения операций ввода-вывода в операционной системе МОС ВП можно использовать два базовых метода:

1) систему управления данными (СУД-32) операционной системы МОС ВП;

2) системные программы обслуживания ввода-вывода.

Система управления данными (СУД-32) предоставляет ряд программ для выполнения независящих от устройства функций общего назначения, таких как хранение данных, их поиск и модификация.

Системные программы обслуживания ввода-вывода позволяют непосредственно использовать ресурсы ввода-вывода операционной системы МОС ВП методом, зависящим от конкретных устройств. Программы обслуживания ввода-вывода также предоставляют некоторые специальные функции, недоступные в СУД-32. Использование программ обслуживания ввода-вывода требует от пользователя больше знаний, чем если бы он использовал СУД-32, но приводит к более эффективным операциям ввода-вывода.

Системные программы обслуживания ввода-вывода:

1) назначить канал ввода-вывода (¤ASSIGN);

2) освободить канал (¤DASSGN);

3) поставить в очередь запрос на ввод-вывод (¤QIO);

4) поставить в очередь запрос на ввод-вывод и ждать флага событий (¤QIOW);

5) форматировать вход (¤FAO);

6) форматировать вход с параметром (¤FAOL);

7) распределить устройство (¤ALLOC);

8) освободить устройство (¤DALLOC);

9) смонтировать том (¤MOUNT);

10) демонтировать том (¤DISMOU);

11) получить информацию об устройстве и канале (¤GETDVI);

12) получить информацию об устройстве и канале и ждать (¤GETDVIW);

13) отменить операцию ввода-вывода в канале (¤CANCEL);

14) создать почтовый ящик и назначить канал (¤CREMBX);

15) удалить почтовый ящик (¤DELMBX);

16) разослать сообщение (¤BRKTHRU);

17) разослать сообщение и ждать (¤BRKTHRUW);

18) послать сообщение контроллеру заданий (¤SNDJBC);

19) послать сообщение контроллеру заданий и ждать (¤SNDJBCW);

20) послать сообщение оператору (¤SNDOPR);

21) послать сообщение в файл регистрации ошибок (¤SNDERR);

22) получить сообщение (¤GETMSG);

23) выдать сообщение (¤PUTMSG).

В разделе 7 приводится общая информация о том, как использовать программы обслуживания ввода-вывода, включая:

1) назначение каналов;

2) постановка в очередь запросов на ввод-вывод;

3) распределение устройств;

4) использование почтовых ящиков.

Приведены примеры, показывающие, как использовать программы обслуживания ввода-вывода с терминала (см. Документ [2]).

7.1. Назначение каналов

Перед тем как могут быть выполнены операции ввода-вывода на физическом устройстве, этому устройству должен быть назначен канал, чтобы обеспечить путь между процессом и устройством. Программа системного обслуживания устанавливает этот путь.

При вызове служебной программы #ASSIGN необходимо предоставить имя устройства, которое может быть именем физического устройства или логическим именем, и адрес слова, в которое заносится номер канала. Служебная программа возвращает номер канала, который программист использует в своей программе, когда он пишет запрос на ввод-вывод.

В приведенном примере устройству TTA2 назначается канал ввода-вывода. Номер канала возвращается в слове TTCHAN.

Пример.

```
TTNAME: .ASCID  /TTA2:/          дескриптор терминала
TTCHAN: .BLKW   1                номер канала терминала
```

```
DEVNAM = TTNAME,-
CHAN   = TTCHAN
```

Чтобы назначить канал текущему устройству ввода-вывода по умолчанию, следует использовать логическое имя SYS$INPUT или SYS$OUPUT (подраздел 7.10).

7.2. Постановка в очередь запросов на ввод-вывод

Все операции ввода-вывода в операционной системе мос вп инициируются программой системного обслуживания $QIO. Программа $QIO ставит в очередь запрос и возвращает управление. Пока операционная система МОС ВП обрабатывает этот запрос, программа, которая выдала запрос, может продолжать выполнение.

Аргументы, которые требуются для служебной программы $QIO, включают номер канала, назначенный устройству, на котором должны выполняться операции ввода-вывода, и код функции (выраженный символически), который указывает, какие специфические операции должны быть выполнены. В зависимости от кода функции могут потребоваться от одного до шести дополнительных параметров.

Например, коды функций IO$_WRITEVBLK и IO$_READVBLK не зависят от устройств и используются для чтения и записи отдельных записей или виртуальных блоков. Эти коды функций удобны для простого ввода-вывода с терминала. Они требуют параметров, указывающих адрес входного или выходного буфера и длину буфера. Вызов программы $QIO для записи строки на

терминале приведен в примере.

Пример.

```
¤QIO_S  CHAN=TTCHAN, -
        FUNC=#IO¤_WRITEVBLK,-
        P1=BUFADDR,-
        P2=#BUFLEN
```

Коды функций определены для всех поддерживаемых типов устройств, и большинство кодов являются зависимыми от устройств, т.е. они выполняют функции, специфичные для конкретного устройстза. Макрокоманда ¤IODEF определяет символические имена для этих кодов функций.

7.3. Синхронизация завершения служебной программы

Программы системного обслуживания ¤ENQ и ¤QIO возвращают управление вызывающей программе, как только запрос поставлен в очередь. Код состояния возвращается в регистре R0 и указывает, успешно или неудачно запрос был поставлен в очередь. Чтобы гарантировать правильную синхронизацию операции постановки в очередь по отношению к программе, последняя должна выполнить следующее:

1) проверить, что операция успешно постановлена в очередь;

2) проверить, что сама операция успешно закончилась.

Необязательные аргументы служебных программ ¤ENQ и QIO обеспечивают методы синхронизации завершения ввода-вывода. Имеется три метода, которые можно использовать для проверки

завершения управления захватом или запроса на ввод-вывод:

1) указать номер флага события, который необходимо установить после завершения операции;

2) указать адрес программы AST, которая должна выполняться после завершения операции;

3) указать адрес блока состояния ввода-вывода, или блока состояния захвата, в который операционная система мос вл. может поместить состояние возврата после завершения операции.

Примеры 1, 2, 3 иллюстрируют использование этих трех методов.

Пример 1.

```
$QIO_S  EFN=#1,...          выдать первый запрос на ввод-
                            вывод
BSBW    ERROR               успешно поставлен в очередь?
$QIO_S  EFN=#2,...          ; выдать второй запрос на ввод-
                            вывод
BSBW    ERROR               успешно поставлен в очередь?
$WFLAND_S -                 ждать пока выполняются оба
        EFN=#0, -
        MASK=#^B110
```

Примечания:

1. Если в качестве аргумента указывается номер флага события, программа $QIO сбрасывает этот флаг, когда ставит в очередь запрос на ввод-вывод. Когда ввод-вывод заканчивается, флаг устанавливается.

2. В приведенном примере программа выдает два требова-

ния на постановку в очередь запросов на ввод-вывод. Для каждого из этих запросов указаны различные флаги событий.

3. Программа системного обслуживания $WFLAND ("ждать логического "и" флагов событий") помещает процесс в состояние ожидания, пока не закончатся обе операции ввода-вывода. Аргумент EFN указывает, что оба флага находятся в кластере 0. Аргумент MASK указывает, какие флаги ожидаются.

4. Программа системного обслуживания $WFLAND (и другие программы системного обслуживания, связанные с ожиданием) ожидает установки флага события, а не завершения операции ввода-вывода. Если бы какое-либо другое событие установило требуемые флаги событий, то ожидание флага событий закончилось бы преждевременно. Использование флагов событий должно быть четко скоординировано. (п. 7.3.1)

Пример 2.

```
$QIO_S  ...,ASTADR=TTAST, -
            ASTPRM=#1,
BSBW    ERROR                       успешно поставлен в
                                    очередь?


.ENTRY          TTAST,^M<R10,R11>   маска входа для под-
                                    программы обслуживания
                                    AST
                                    обработка завершения
                                    ввода-вывода
```

```
RET                                     конец подпрограммы
                                        обслуживания AST
```

Примечания:

1. Если задан аргумент ASTADR при вызове программы системного обслуживания $QIO, то операционная система мос вл. прерывает процесс, когда завершается ввод-вывод, и передает управление указанной программе обслуживания AST.

2. При вызове программы системного обслуживания $QIO задается адрес подпрограммы AST и параметр, который передается в качестве аргумента подпрограмме обслуживания AST. Когда программа $QIO возвращает управление, процесс продолжает выполнение.

3. Когда завершается ввод-вывод, вызывается подпрограмма AST с именем TTAST, которая резгирует на завершение ввода-вывода. Проверяя параметр, пожпрограмма AST может определить источник запроса на ввод-вывод.

Когда эта подпрограмма заканчивает выполнение, управление передается процессу в ту точку, в которой он был прерван. Если при вызове программы $QIO задан аргумент ASTADR, то следует также указать аргумент IOSB, чтобы подпрограмма AST могла оценить, успешно ли закончилась операция ввода-вывода.

Пример 3.

```
TTIOSB: .BLKQ   1                       блок состояния
                                        ввода-вывода
```

```
        $QIO_S  ...,IOSB=TTIOSB,...     выдать запрос
                                        на ввод-вывод
        BSBW    ERROR                   успешно поставлен
                                        в очередь?
                                        продолжить

10$:    TSTW    TTIOSB                  операция ввода-вывода уже
                                        выполнена?
        BEQL    10$                     если нет, то выполнять
                                        проверку, пока не выпол-
                                        нится
        SMPW    TTIOSB,#SS$_NORMAL ; операция ввода-вывода закончилась
                                        успешно?
        BNEQ    IO_ERR                  если нет, то обработать
                                        ошибку
```

Примечания:

1. Блок состояния ввода-вывода является структурой размером в квадрослово, которую операционная система МОС ВП использует для извещения о состоянии операции ввода-вывода. Эту область размером в квадрослово необходимо определить в программе пользователя.

2. TTIOSB определяет блок состояния ввода-вывода для этой операции ввода-вывода. Аргумент IOSB в программе системного обслуживания $QIO ссылается на это квадрослово.

3. Программа $QIO ожидает это квадрослово, когда она ставит в очередь запрос на ввод-вывод. После того как запрос поставлен в очередь, программа вызывает подпрограмму, чтобы проверить, успешно ли поставлен запрос в очередь. Если да, программа продолжает выполнение.

4. Процесс опрашивает блок состояния ввода-вывода. Если младшее слово его содержит 0, то это означает, что операция ввода-вывода еще не закончилась. В примере 3 программа выполняет цикл, пока запрос не завершится.

5. Как только операция ввода-вывода заканчивается, процесс сравнивает младшее слово блока состояния ввода-вывода с кодом успешного состояния SS$_NORMAL. Если состояние возврата не совпадает с SS$_NORMAL, то программа переходит к обработке ошибки IO_ERR.

6. Метод, показанный в примере 3, впустую тратит системное время, выполняя цикл, пока не закончится запрос. Этот метод следует использовать, только если он является последней возможной альтернативой.

7.3.1. Рекомендуемый метод для проверки асинхронного завершения

Для определения завершения асинхронного события рекомендуется использовать программу системного обслуживания $SYNCH. ("синхронизировать"). Служебная программа $SYNCH корректно ожидает действительного завершения асинхронного события, даже если какое-то другое событие устанавливает флаг события.

Чтобы использовать служебную программу ¤SYNCH для ожидания завершения асинхронного события, необходимо указать как номер флага события, так и адрес блока состояния ввода-вывода (IOSB) при вызове программы системного обслуживания. Асинхронная служебная программа ставит запрос в очередь и возвращает управление в программу. Когда асинхронная служебная программа завершает выполнение, она устанавливает флаг события и помещает код конечного состояния запроса в блок IOSB.

При вызове программы ¤SYNCH, необходимо задать тот же аргумент EFN и блок состояния ввода-вывода, которые были указаны при вызове асинхронной служебной программы. Программа ¤SYNCH ожидает флага события, который должен быть установлен с помощью программы системного обслуживания ¤WAITFR. Когда устанавливается указанный флаг события, программа ¤SYNCH проверяет заданный блок состояния ввода-вывода.

Если блок состояния ввода-вывода ненулевой, то программа системного обслуживания завершилась и программа ¤SYNCH возвращает управление в программу пользователя. Если блок состояния ввода-вывода содержит нуль, то программа ¤SYNCH сбрасывает флаг события с помощью служебной программы ¤CLREF и вызывает служебную программу ¤WFLOR для ожидания установки флага события.

Программа ¤SYNCH устанавливает флаг события и возвращает управление в программу. Это дает гарантию, что вызову программы ¤SYNCH не помешает проверка завершения другого

асинхронного события, которое завершается приблизительно в то же время и использует тот же флаг события для сигнализации о завершении.

Пример.

```
EVENT_FLAG = 1

Q_IOSB: .QUAD   0


        $QIO_S  EFN=#EVENT_FLAG,-
                IOSB=Q_IOSB,...
        $SYNCH_S, -
                EFN=#EVENT_FLAG
                IOSB=Q_IOSB
        BLBC    R0,ERROR
```

Примечание. Служебная программа $QIOW заменяет комбинацию процедур $QIO и $SYNCH. В приведенном фрагменте программы показан пример работы программы $SYNCH.

7.4. Синхронные формы программ обслуживания ввода-вывода

Некоторые программы обслуживания ввода-вывода можно выполнять либо синхронно, либо асинхронно. Символ "W" в

конце имени программы системного обслуживания определяет синхронную версию служебной программы.

Синхронная версия программы системного обслуживания комбинирует функции асинхронной версии этой программы и программы системного обслуживания ¤SYNCH. Синхронная версия работает так, как если бы использовалась асинхронная версия программы системного обслуживания, за которой сразу бы следовал вызов программы ¤SYNCH. Она помещает запрос на ввод-вывод в очередь и затем переводит программу в состояние ожидания, пока не завершится запрос на ввод-вывод. Синхронная версия требует те же аргументы, что и асинхронная версия.

В табл. 31 приводится список имен асинхронных и синхронных программ обслуживания ввода-вывода.

Таблица 31

Асинхронные программы	Синхронные программы	Описание
¤BRKTHRU	¤BRKTHRUW	Разослать сообщение
¤GETDVI	¤GETDVIW	Получить информацию об устройстве или томе
¤GETJPI	¤GETJPIW	Получить информацию о задании или процессе
¤GETLKI	¤GETLKIW	Получить информацию о захвате
¤GETQUI	¤GETQUIW	Получить информацию об очереди

Продолжение табл. 31

Асинхронные программы	Синхронные программы	Описание
$QIO	$QIOW	Поставить в очередь запрос на ввод-вывод
$SNDJBC	$SNDJBCW	Послать сообщение контроллеру заданий
$UPDSEC	$UPDSECW	Обновить файл секции на диске

7.5. При завершении операции ввода-вывода, операционная система МОС ВП извещает о состоянии завершения в блоке состояния ввода-вывода, если он специфицирован. Состояние завершения сообщает закончилась ли операция успешно или нет, число переданных байтов и дополнительную информацию возврата, зависящую от устройства.

На рис. 39 показан формат информации, записанной в блок IOSB.

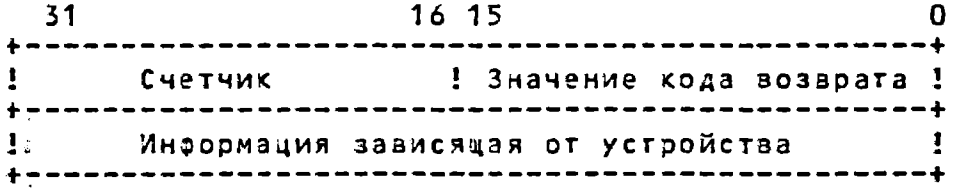


Рис. 39

Первое слово содержит системный код состояния, индицирующий успех или неудачу операции. Используемые коды сос-

тояния совпадают с теми, которые возвращают все программы системного обслуживания, например, код SS¤_NORMAL свидетельствует об успешном завершении.

Второе слово содержит число байтов, которое реально передано в операции ввода-вывода (см. докумнет [5]).

Второе длинное слово содержит информацию возврата, зависящую от устройства.

Чтобы удостовериться в успешном завершении операции ввода-вывода и целостности переданных данных, блок IOSB следует проверять сразу после запросов на ввод-вывод, в особенности для функций ввода-вывода, зависящих от устройств. (см. документ [2]).

7.6. Освобождение каналов ввода-вывода

Если процесс больше не нуждается в доступе к устройству ввода-вывода, ему следует освободить назначенный устройству канал. Это делается с помощью вызова программы системного обслуживания ¤DASSGN.

Пример.

```
¤DASSGN_S CHAN=TTCHAN
```

Операционная система МОС ВП автоматически освобождает канал для процесса, когда образ, для которого был назначен канал, завершает свою работу.

Пример иллюстрирует полную последовательность операций ввода-вывода, использующих макрокоманду ¤QIOW для чтения и записи строк на текущее устройство по умолчанию с именем SYS¤INPUT. Эта программа будет работать корректно, только

если она выполняется интерактивно, поскольку ввод-вывод должен выполняться на текущем терминале.

Пример.

```
TTNAME: .ASCID  /SYS$INPUT:/        дескриптор для имени
                                    терминала
TTCHAN: .BLKW   1                   получить здесь номер
                                    канала
TTIOSB: .BLKW   1                   первое слово блока IOSB,
                                    состояние
TTIOLEN:
        .BLKW   1                   второе слово, получить
                                    длину
        .BLKL   1                   второе длинное слово
                                    блока IOSB
OUTLEN: .BLKL   1                   длина строки для выхода
INBUF:  .BLKB   80                  буфер для чтения входа


10$:    $ASSIGN_S -                 назначить канал
                DEVNAM=TTNAME, -;   логическое имя, трансли-
                CHAN=TTCHAN         руемое программой $ASSIGN
        BSBW    ERROR
        $QIOW_S -
                FUNC=#IO$_READVBLK -
                CHAN=TTCHAN, -
                LENGTH=P2, -
```

```
                BUFFER=P1, -
                IOSB=TTIOSB
        BSBW    ERROR
        MOVZWL  TTIOSB,R0           поместить код состояния
                                    в регистр R0
    BSBW    ERROR
    MOVZWL  TTIOLEN,OUTLEN          получить длину из блока
                                    IOSB
      $QIOW_S -
                FUNC=#IO$_WRITVBLK -
                CHAN=TTCHAN, -
                LENGTH=P2, -
                BUFFER=P1, -
                IOSB=TTIOSB
    BSBW    ERROR
      MOVZWL  TTIOSB,R0             поместить код состояния
                                    в регистр R0
      BSBW    ERROR
    $DASSNG_S -                     выполнено, освободить
                                    канал
                CHAN=TTCHAN
      BSBW    ERROR

ERROR:  BLBS    R0,10$              проверить, успешный ли
                                    код возврата
```

```
        ¤EXIT_S RO                      если нет, то выйти и
                                        просигнализировать
10¤:    RSB                             если успешный, вернуть-
                                        ся в точку вызова
```

Примечания:

1. TTNAME - это дескриптор строки символов для логического устройства SYS¤INPUT, TTCHAN - это слово, в которое заносится номер канала, назначенный этому устройству.

2. Блок IOSB для операций ввода-вывода имеет такую структуру, чтобы программа легко могла проверить состояние завершения (в первом слове) и длину возвращаемой входной строки (во втором слове).

3. Строка будет читаться в буфер INBUF. Длинное слово OUTLEN будет содержать длину строки для операции ввода-вывода.

4. Программа ¤ASSIGN назначает канал и пишет номер канала в TTCHAN.

5. Если служебная программа ¤ASSIGN завершается успешно, то программа ¤QIOW читает строку с терминала и требует, чтобы в блок состояния ввода-вывода с именем TTIOSB пришло извещение о состоянии завершения.

6. Процесс ожидает, пока закончится ввод-вывод, затем проверяет в первом слове блока состояния ввода-вывода состояния возврата. Если состояние не успешное, то программа переходит на обработку ошибки.

7. Затем в длинное слово OUTLEN помещается длина прочитанной строки. Это необходимо, поскольку программа ¤QIOW

требует аргумента размером в длинное слово, а поле длины в блоке состояния ввода-вывода имеет размер слово. Программа ¤QIOW пишет только что прочитанную строку на терминал.

8. Программа выполняет контроль ошибок: сначала убеждается, что успешно поставила в очередь запрос на ввод-вывод, затем после завершения запроса она проверяет, успешно ли закончилась операция ввода-вывода.

9. После того как все операции ввода-вывода завершаются, канал освобождается.

7.7. Отмена запросов на ввод-вывод

Если процесс должен отменить запросы на ввод-вывод, которые были поставлены в очередь, но еще не успели выполниться, он может вызвать программу системного обслуживания ¤CANCEL ("отменить ввод-вывод в канале"). При этом будут отменены все запросы на ввод-вывод этого процесса, находящиеся в очереди к данному каналу. Конкретный запрос на ввод-вывод указать нельзя.

Программу системного обслуживания ¤CANCEL можно вызвать слудующим образом:

```
¤CANCEL_S CHAN=TTCHAN
```

В приведенном примере программа системного обслуживания ¤CANCEL инициирует отмену всех запросов на ввод-вывод, ожидающих в канале, чей номер помещен в TTCHAN.

Программа системного обслуживания ¤CANCEL возвращает управление после инициирования отмен запросов на ввод-вывод. Если при вызове ¤QIO указан флаг события, подп-

программа обслуживания AST или блок состояния ввода-вывода, то операционная система МОС ВП устанавливает флаг, доставляет AST или извещает блок состояния ввода-вывода, соответственно, после реального завершения отмены запросов.

7.8. Распределение устройств

Многие устройства ввода-вывода являются разделяемыми, т.е. одновременно к такому устройству могут иметь доступ несколько процессов. Вызывая программу системного обслуживания ¤ASSIGN, процесс получает канал к устройству для выполнения операций ввода-вывода.

В некоторых случаях процессу может понадобиться монопольное использование устройства так, чтобы другим процессам устройство было недоступно. Чтобы зарезервировать устройство для монопольного использования, его необходимо распределить.

Распределение устройства обычно выполняется из потока команд языка DCL командой ALLOCATE. Процесс может также распределить устройство, вызывая программу системного обслуживания ¤ALLOC ("распределить устройство"). Когда устройство распределяется процессом, только процесс, распределяющий устройство, и все созданные им подпроцессы могут назначать каналы к устройству.

При вызове программы системного обслуживания ¤ALLOC необходимо задать имя устройства. Задаваемое имя устройства может быть одного их трех типов:

1) именем физического устройства, например лентопротяжное устройство MT33;

2) логическим именем, например, TAPE;

3) обобщенным именем устройства, например, мт.

Если указывается имя физического устройства, то программа ¤ALLOC пытается распределить заданное устройство.

Если указывается логическое имя, то программа ¤ALLOC транслирует логическое имя и пытается распределить имя физического устройства, приравненное логическому имени.

Если указывается обобщенное имя устройства, т.е. если указывается тип устройства, но не указывается контроллер и/или номер устройства, программа ¤ALLOC пытается распределить любое доступное устройство указанного типа.

Если задаются обобщенные имена устройств, необходимо обеспечить поля в которые программа системного обслуживания ¤ALLOC возвращает имя и длину физического устройства, которое реально распределяется, так что это имя можно использовать в качестве входа для программы системного обслуживания ¤ASSIGN.

Пример иллюстрирует распределение устройства управления лентами, которое задается логическим именем TAPE.

Пример.

```
LOGDEV: .ASCID  /TAPE/      дескриптор для логического
                            имени
DEVDESC:                    дескриптор для физического
                            имени
        .LONG   64          длина буфера
```

```
        .ADDRESS -          адрес буфера
                DEVSTR
DEVSTR: .BLKB   64          получить возвращенное фи-
                            зическое имя
TAPECHAN:
        .BLKW   1                   канал для ввода-
                                    вывода с ленты


        $ALLOC_S -
                DEVNAM=LOGDEV,-
                PHYLEN=DEVDESC,-
                PHUYUF=DEVDESC
        BSBW    ERROR
        $ASSIGN_S -                   назначить канал
                DEVNAM=DEVDESC,-
                CHAN=TAPECHAN
        BSBW    ERROR
                                    продолжить ввод-вывод


        $DASSGN_S -                 освободить канал
                CHAN=TAPECHAN
        BSBW    ERROR
        $DALLOC_S -                 освободить ленту
                DEVNAM=DEVDESC
```

Примечания:

1. Служебная программа ¤ALLOC запрашивает распределение устройства, соответствующего логическому имени TAPE, определенному дескриптором строки LOGDEV. Аргумент DEVDESC ссылается на буфер, куда записывается имя физического устройства, которое реально распределяется и длина строки имени. Программа ¤ALLOC транслирует логическое имя tape и возвращает строку эквивалентного имени устройства, которое реально распределено, в буфер DEVDESC. Программа записывает длину строки в первое слово буфера DEVDESC.

2. Программа ¤ASSIGN использует строку символов, возвращаемую программой системного обслуживания ¤ALLOC в качестве входного аргумента имени устройства и требует, чтобы номер канала был записан в TAPECHAN.

3. После завершения операций ввода-вывода программа системного обслуживания ¤DASSGN освобождает канал, а программа системного обслуживания ¤DALLOC обсвобождает устройство. Канал должен быть освобожден раньше, чем устройство.

7.8.1. Неявное распределение

Устройства, которые не могут разделяться несколькими процессами (например, терминалы и построчно-печатающее устройство) не нуждаются в явном распределении. Поскольку эти устройства неразделяемые, они распределяются неявно программой системного обслуживания ¤ASSIGN, когда эта программа вызывается для назначения канала устройству.

7.8.2. Освобождение устройства

Когда завершается программа, использующая распределенное устройство, желательно, чтобы она освободила устройство с помощью программы системного обслуживания ¤DALLOC ("освободить устройство"), с целью сделать его доступным для других процессов, как показано в примере.

Пример.

```
¤DALLOC_S DEVNAM=DEVDESC
```

При завершении работы образа операционная система мос вп. автоматически освобождает устройства, распределенные образу.

7.9. Монтирование и демонтирование томов

Монтирование тома устанавливает связь между томом, устройством и процессом. Чтобы можно было выполнить операции ввода-вывода на томе, этот том или группу томов необходимо смонтировать. Обычно пользователь монтирует или демонтирует том из потока команд DCL с помощью команд MOUNT или DISMOUNT. Процесс может также монтировать том или ряд томов, используя программу системного обслуживания ¤MOUNT ("монтировать том"). Том или группу томов можно демонтировать с помощью программы системного обслуживания ¤DISMOU ("демонтировать том").

Монтирование тома включает две отдельные операции:

1) поместить том на устройство и подготовить к работе это устройство;

2) смонтировать том с помощью программы системного обслуживания ¤MOUNT.

7.9.1. Вызов программы системного обслуживания ¤MOUNT

Программа системного обслуживания ¤MOUNT позволяет процессу смонтировать либо отдельный том, либо группу томов. При вызове программы системного обслуживания ¤MOUNT необходимо указать имя устройства.

Программа системного обслуживания ¤MOUNT имеет единственный аргумент - адрес списка дескрипторов элементов. Этот список завершается длинным словом из двоичных нулей. На рис. 40 показан формат дескриптора элемента.

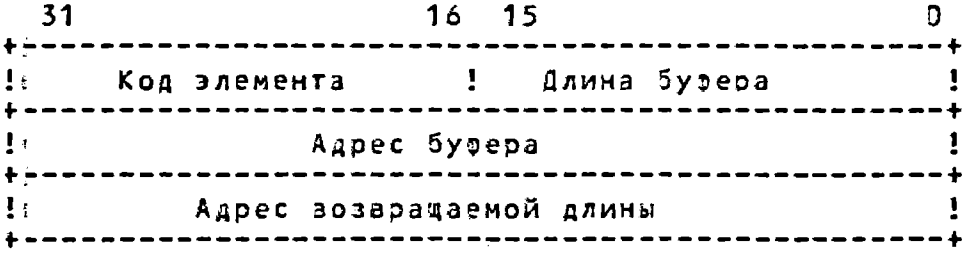


Рис. 40

Большинство дескрипторов элементов не должны задаваться в каком-либо порядке. Чтобы смонтировать набор томов, необходимо указать один дескриптор элемента на устройство и один дескриптор элемента на том. Дескрипторы для томов должны указываться в том же порядке, что и дескрипторы для устройств на которые они загружаются. Для дескрипторов элементов, отличных от имени устройства или тома, если указы-

вается один и тот же дескриптор более одного раза, то используется последняя из ссылок на этот дескриптор.

Пример иллюстрирует вызов программы ¤MOUNT. Этот вызов эквивалентен следующей команде языка DCL:

¤MOUNT/SYSTEM/NOQUOTA DRA4:,DRA5: USER01,USERD2,USERD¤

Пример.

```
        ¤MNTDEF
ITEMS:  .WORD   4               длина флагов
        .WORD   MNT¤_FLAGS      код флага
        .ADDRESS -              адрес длинного слова
                FLAGS           флага
        .LONG   0               неиспользуемое длинное
                                слово
;
        .WORD   5               длина имени первого
                                устройства
        .WORD   MNT¤_DEVNAM     код устройства
        .ADDRESS -              адрес имени первого
                DEV1            устройства
        .LONG   0               неиспользуемое длинное
                                слово

        .WORD   6               длина имени первого тома
        .WORD   MNT¤_VOLNAM     код тома
        .ADDRESS -              адрес имени первого тома
                VOL1
        .LONG   0               неиспользуемое длинное
```

```
                                        слово

        .WORD   5                       длина имени второго уст-
                                        ройства
        .WORD   MNT$_DEVNAM             код устройства
        .ADDRESS -                      адрес имени второго уст-
                DEV2                    ройства
        .LONG   0                       неиспользуемое длинное
                                        слово
;
        .WORD   6                       длина имени второго тома
        .WORD   MNT$_VOLNAM             код тома
        .ADDRESS -
                VOL2                    адрес имени второго тома
        .LONG   0                       неиспользуемое длинное
                                        слово

        .WORD   6                       длина логического имени
                                        тома
        .WORD   MNT$_LOGNAM             код логического имени
        .ADDRESS -                      адрес логического имени
                LOG                     тома
        .LONG   0                       неиспользуемое длинное
                                        слово
        .LONG   0                       конец элементного списка

DEV1:   .ASCII  /DRA4:/                 первое устройство
```

```
VOL1:   .ASCII  /USER01/          имя первого тома
DEV2:   .ASCII  /DRA5:/           второе устройство
VOL2:   .ASCII  /USER02/          имя второго устройства
LOG:    .ASCII  /USERD¤/          логическое имя

FLAGS:  .LONG   <MNT¤M_SYSTEM!MNT¤M_NODISKQ>
```

¤MOUNT_S - теперь вызвать ¤MOUNT ITMLST=ITEMS

7.9.2. Вызов программы системного обслуживания ¤DISMOU

Программа системного обслуживания ¤DISMOU позволяет процессу демонтировать один том или группу томов. При вызове программы ¤DISMOU необходимо указать имя устройства. Если том, смонтированный на данном устройстве, является частью смонтированной группы томов и не заданы никакие флаги, то демонтируется вся группа томов.

Пример иллюстрирует вызов программы ¤DISMOU. При работе программы демонтируется группа томов, смонтированная в примере, показанном в п. 7.9.1.

пример.

```
DEV1_DESC:
        .ASCID  /DRA4:/



        $DISMOU_S -
                DEVNAM=DEV1_DESC
```

7.10. Логические имена и имена физических устройств

Когда имя устройства указывается в качестве входного для программы системного обслуживания ввода-вывода, оно может быть либо именем физического устройства, либо логическим именем. Если имя устройства содержит двоеточие, то двоеточие и все следующие за ним символы игнорируются. Если строке имени устройства предшествует символ подчеркивания, то это означает, что данная строка является именем физического устройства.

Пример.

```
TTNAME:.ASCID /_TTB3:/
```

Любая строка, которая начинается не со знака подчеркивания, рассматривается как логическое имя, даже если это

физическое имя устройства. Следующие программы системного обслуживания транслируют логическое имя итеративно, пока не будет возвращено имя физического устройства, или пока не будет выполнено такое количество трансляций, которое задано по умолчанию:

1) распределить устройство (¤ALLOC);

2) назначить канал ввода-вывода (¤ASSIGN);

3) разослать сообщение (¤BRKTHRU);

4) освободить устройство (¤DALLOC);

5) демонтировать том (¤DISMOU);

6) получить информацию об устройстве/томе (¤GETDVI);

7) монтировать том (¤MOUNT).

При каждой трансляции по порядку просматриваются таблицы логических имен, определенные логическим именем LNM¤FILE_DEV. Это обычно таблицы LNM¤PROCESS, LNM¤JOB, LNM¤GROUP и LNM¤SYSTEM, перечисленные в порядке просмотра. Если в результате трансляции определяется физическое имя устройства, то для этого устройства выполняется запрос на ввод-вывод.

Если служебные програмы не находят в таблице логических имен элемент для логического имени, то программа обслуживания ввода-вывода рассматривает указанное имя в качестве имени физического устройства. Если при вызове одной из этих служебных программ указывается имя реального физического устройства, то следует ввести символ подчеркивания, чтобы избежать трансляции логического имени.

Когда программа системного обслуживания ¤ALLOC возвра-

щает имя распределенного физического устройства, возвращаемая строка с именем имеет префикс в виде символа подчеркивания. Когда это имя используется для последующего вызова программы системного обслуживания $ASSIGN, эта программа не пытается транслировать имя устройства.

Если при вызове программ обслуживания ввода-вывода используется логическое имя, то перед выполнением программы необходимо установить правильную эквивалентность имени устройства. Это можно сделать либо с помощью команды DEFINE языка DCL, либо непосредственно в программе перед обращением к программе обслуживания ввода-вывода с помощью вызова программы системного обслуживания $CRELNM ("создать логическое имя"), которая устанавливает эквивалентное имя (см. раздел 6).

7.10.1. Имена устройств, принятые по умолчанию

Если после трансляции логического имени строка с именем устройства при вызове программы системного обслуживания ввода-вывода не полностью специфицирует имя устройства (то есть тип устройства, контроллер и устройство), то эта программа дополняет незаданные поля, либо значениями, принятыми по умолчанию, либо значениями, выбранными по принципу доступности устройства.

Используются следующие правила:

1) программы системного обслуживания $ASSIGN и $DALLOC используют значения, принятые по умолчанию из табл. 32;

2) программа системного обслуживания ¤ALLOC рассматривает имя устройства как обобщенное имя и пытается найти устройство, которое удовлетворяет компонентам заданного имени устройства (см. табл. 32).

7.11. Получение информации о физических устройствах

Программа ¤GETDVI возвращает информацию об устройствах. Возвращаемая информация задается списком элементов, который создается перед вызовом программы ¤GETDVI.

При вызове программы системного обслуживания ¤GETDVI, необходимо предоставить адрес списка элементов, который задает, какая информация должна быть возвращена (см. документ [2]). В тех случаях, когда при вызове программы обслуживания ввода-вывода используется обобщенное имя (т.е. неспецифическое) устройства, программе может понадобиться узнать, какое именно устройство было реально использовано. Чтобы выяснить это, программа должна передать программе ¤GETDVI номер канала для устройства и затребовать имя устройства в элементном идентификаторе DVI¤_DEVNAM.

Таблица 32

Имена устройств, принимаемые по умолчанию для системных программ обслуживания ввода-вывода

Заданное имя устройства	Имя устройства по умолчанию для большинства программ обслуживания ввода-вывода	Обобщенные имена устройств, используемые программами ^ASSIGN и ^DALLOC
DD:	DRAD: (устройство 0 на контроллере a)	DDXY: (любое доступное устройство указанного типа)
DDC:	DDC0: (устройство 0 на указанном контроллере)	DDCY: (любое доступное устройство на указанном контроллере)
DDU:	DDAU: (указанное устройство на контроллере a)	DDXU: (устройство указанного типа на любом доступном контроллере)
DDCU:	DDCU: (указанные устройство и контроллер)	DDCU: (указанные устройство и контроллер)

DD: - заданный тип устройства;

C: - заданный контроллер;

X: - любой контроллер;

U: - заданный номер устройства;

Y: - любой номер устройства.

7.12. Форматирование выходных строк

При подготовке выходных строк для программ может понадобиться перед выводом ввести в строку переменную информацию или преобразовать числовое значение в строку в коде КОИ-8. Эти функции выполняет программа системного обслуживания ¤FAO.

Служебная программа состоит из следующих компонент:

1) управляющей строки, которая содержит фиксированную часть с текстом и директив форматирования. Директивы определяют позицию внутри строки, где должны быть сделаны подстановки и описывают тип данных и длину входных величин, которые должны быть заменены или преобразованы;

2) выходного буфера, который должен содержать строку после преобразований и подстановок;

3) необязательного аргумента, указывающего слово, в которое записывается конечная длина форматированной выходной строки;

4) параметров, которые обеспечивают аргументы для директив форматирования.

Пример показывает вызов программы системного обслуживания ¤FAO с целью форматирования выходной строки для макрокоманды ¤QIOW.

Пример.

```
FAOSTR: .ASCID  /файл !AS  не существует/  дескриптор для
                                           управляющей строки
                                           программы ¤FAO
FAODESC:                                   дескриптор для вы-
```

```
                                        хода программы ¤FAO
        .LONG   80                      длина буфера
        .ADDRESS -                      адрес буфера
                FAOBUF
FAOBUF: .BLKB   80                      буфер для выхода
                                        программы ¤FAO
FAOLEN: .LONG   0                       получить длину вы-
                                        хода программы
                                        ¤FAO
FILESPEC:
        .ASCID  /DISK¤USER:MYFILE.DAT/; дескриптор для па-
                                        раметра программы
                                        ¤FAO

        ¤FAO_S  CTRSTR=FAOSTR, -
                OUTLEN=FAOLEN, -
                OUTBUF=FAODESC, -
                P1=#FILESPEC            параметр для ¤FAO
        BSBW    ERROR
        ¤QIOW   ...,BUFFER=FAOBUF, -
                LENGTH=FAOLEN
        BSBW    ERROR
```

Примечания:

1. FAOSTR представляет управляющую строку программы ¤FAO. !AS является примером директивы программы FAO. Она требует входного параметра, который задает адрес дескриптора строки символов. Когда программа ¤FAO вызывается для

формирования этой управляющей строки, !AS заменяется строкой, для которой указан адрес дескриптора.

2. FAODESC является дескриптором строки символов выходного буфера. Программа ¤FAO будет писать в этот буфер выходную строку, а длину окончательно сформированной строки запишет в младшее слово длинного слова FAOLEN. (длинное слово резервируется, чтобы использовать его для входного аргумента программы ¤QIOW).

3. FILESPEC - это дескриптор строки символов, определяющий входную строку для директивы !AS программы ¤FAO.

4. Вызов программы ¤FAO задает управляющую строку, поля выходного буфера и длины и параметр P1, являющийся адресом дескриптора строки, которая должна подставляться на место директивы.

5. Когда программа ¤FAO завершается, программа ¤QIOW записывает следующую выходную строку:

файл DISK¤USER:MYFILE.DAT не существует

7.13. Почтовые ящики

Почтовые ящики - это виртуальные устройства, которые используются для связи между процессами. Реальная передача данных выполняется с помощью системы управления данными (СУД-32) или программ обслуживания ввода-вывода. Когда служебная программа ¤CREMBX создает почтовый ящик, она также назначает для него канал, который используется создающим процессом. После этого другие процессы могут назначить каналы к этому почтовому ящику, используя программы систем-

ного обслуживания ¤CREMBX или ¤ASSIGN. (если почтовый ящик расположен в памяти, разделяемой многими процессорами, то для создания или назначения канала к почтовому ящику первый процесс каждого процессора должен использовать служебную программу ¤CREMBX).

Программа системного обслуживания ¤CREMBX создает почтовый ящик. Эта программа идентифицирует почтовый ящик логическим именем, заданным пользователем, и назначает ему эквивалентное имя. Эквивалентное имя является именем физического устройства в формате MBAN, где N - номер устройства. Эквивалентное имя имеет терминальный атрибут.

Когда другой процесс назначает канал к почтовому ящику с помощью программы системного обслуживания ¤CREMBX или ¤ASSIGN, он может идентифицировать этот почтовый ящик его логическим именем. Процесс может получить имя MBAN, транслируя логическое имя (с помощью программы системного обслуживания ¤TRNLNM), или вызвать программу системного обслуживания ¤GETDVI, чтобы получить номер устройства и имя физического устройства.

Почтовые ящики бывают либо временными, либо постоянными. Для создания временных и постоянных почтовых ящиков пользователь должен иметь привилегии TMPMBX и PRMMBX, соответственно.

Для временного почтового ящика служебная программа ¤CREMBX вводит логическое имя и эквивалентное имя в таблицу логических имен LNM¤TEMPORARY_MAILBOX. Операционная система МОС ВП удаляет временный почтовый ящик, когда больше нет ни

одного назначенного ему канала.

Для постоянного почтового ящика служебная программа ¤CREMBX вводит логическое и эквивалентное имя в таблицу логических имен LNM¤PERMANENT_MAILBOX. Постоянные почтовые ящики продолжают существовать до тех пор, пока не будут специально помечены для удаления с помощью программы системного обслуживания ¤DELMBX.

Почтовый ящик, расположенный в памяти, разделяемой многими процессорами, также удаляется, когда произойдут все следующие события:

1) процессор заново загружается;

2) многовходовая память заново не инициализируется;

3) ни один другой процессор не имеет процессов, назначивших каналы к этому почтовому ящику.

Пример показывает как процессы могут связываться между собой посредством почтового ящика.

Пример.

```
Процесс ORION
MBLOGNAM:
        .ASCID  /GROUP100_MAILBOX/
MBUFLEN = 128
MBUFFER:
        .BLKW   1
MBXCHAN:
        .BLKW   1
MBXIOSB:
        .BLKW   1
```

```
MBLEN:  .BLKW   1
        .BLKL   1
OUTLEN: .BLKL   1


        .
        .ENTRY  ORION,_^M<R2,R3,R4>

        ¤CREMBX__S -
                PRMFLG=_#0, -
                CHAN=MBXCHAN, -
                MAXMSG=#MBUFLEN, -
                BUFQUO=_#384,-
                PROMSK=_#_^X0000, -
                LOGNAM=MBLOGNAM
        BSBW    ERROR
        ¤QIO__S CHAN=MBXCHAN, -
                FUNC=_#IO¤__READVBLK,-
                IOSB=MBXIOSB,-
                ASTADR=MBXAST,-
                P1=MBUFFER,-
                P2=#MBUFLEN
        BSBW    ERROR


        RET
        .ENTRY  MBXAST,_^M<R2,R3,R4>
```

```
        CMPW    MBXIOSB,_#SS$_NORMAL
        BNEQ    ASTERR
        MOVZWL  MBLEN,OUTLEN
        $QIOW_S ...,BUFFER=MBUFFER,-
                LENGTH=OUTLEN,...
        BSBW    ERROR


        RET
PROCESS CYGNUS
MAILBOX:
        .ASCID  /GROUP100_MAILBOX/
MAILCHAN:
        .BLKW   1
OUTBUF: .BLKB   128
OUTLEN: .BLKL   1


        .ENTRY  CYGNUS,_^M<R2,R3,R4>
        $ASSIGN__S -
                DEVNAM=MAILBOX,-
                CHAN=MAILCHAN
        BSBW    ERROR



        $QIOW_S CHAN=MAILCHAN,-
```

00152-01 97 06

```
            BUFFER=OUTBUF,-
            LENGTH=OUTLEN,...
    BSBW    ERROR


    RET
```

Примечания:

1. Процесс ORION создает почтовый ящик и получает номер канала в аргументе MBXCHAN.

Аргумент PRMFLG указывает, что почтовый ящик является временным почтовым ящиком. Логическое имя вводится в таблицу логических имен LNM$TEMPORARY_MAILBOX.

Аргумент MAXMSG ограничивает размер сообщений, которые может принимать почтовый ящик. В этом примере указан размер, совпадающий с размером буфера (MBUFFER), который используется при вызове программы $QIO. Буфер для ввода-вывода из почтового ящика должен иметь размер не меньший, чем размер, заданный в аргументе MAXMSG.

Когда процесс создает временный почтовый ящик, количество системной памяти, которая распределяется для буферизации сообщений, вычитается из квоты буфера процесса. Для указания, какую часть квоты процесса желательно использовать для буферизации сообщений, следует использовать аргумент BUFQUO.

Почтовые ящики являются защищенными устройствами. Задавая маску защиты с помощью аргумента PROMSK, можно ограничить доступ к почтовому ящику. (в этом примере все

биты в маске сброшены, указывая тем самым неограниченный доступ с чтением и записью).

2. Создав почтовый ящик, процесс ORION вызывает программу системного обслуживания $QIO, требуя чтобы его известили об окончании ввода-вывода (т.е. когда почтовый ящик получает сообщение) посредством прерывания AST. Процесс может продолжить выполнение, однако при получении сообщения подпрограмма обслуживания AST прервает выполнение.

3. Когда процесс CYGNUS посылает сообщение в почтовый ящик, доставляется AST и процесс ORION реагирует на сообщение. ORION получает длину сообщения из первого слова блока состояния ввода-вывода с меткой MBXIOSB и помещает его в длинное слово OUTLEN, чтобы иметь возможность передать его макрокоманде $QIOW_S.

4. Процесс CYGNUS назначает канал почтовому ящику, указывая логическое имя, которое процесс ORION дал почтовому ящику. Программа системного обслуживания $QIOW пишет сообщения из выходного буфера с именем OUTBUF.

7.13.1. Формат почтового ящика

Строка логического имени, назначенного почтовому ящику, определяет, где расположен этот почтовый ящик - в памяти, используемой одним процессором, или в памяти, разделяемой многими процессорами. Аргумент LOGNAM служебной программы $CREMBX задает дескриптор, указывающий на строку символов следующего формата:

Имя-разделяемой-памяти: имя-почтового-ящика

где имя-разделяемой-памяти - указывает, что почтовый ящик расположен в названной памяти, разделяемой многими процессорами. Имя этой памяти задается во время генерации операционной системы МОС ВП. Например, строка SHRMEM=1:CHKPNT идентифицирует почтовый ящик с именем CHKPNT, расположенный в разделяемой памяти с именем SHRMEM=1. Если эта часть строки не включена в логическое имя, значит почтовый ящик не расположен в памяти, разделяемой многими процессорами;

имя-почтового-ящика - это имя, назначенное почтовому ящику длиной от 1 до 15 символов.

При желании для почтового ящика, расположенного в памяти, разделяемой многими процессорами, можно включать оба имени - имя-разделяемой-памяти и имя-почтового-ящика. Однако, если желательно использовать существующие программы без повторной компиляции или компоновки, то можно указывать только имя-почтового-ящика, предоставив операционной системе МОС ВП транслировать его до полной спецификации. Операционная система МОС ВП пытается выполнить трансляцию строки, указанной аргументом LOGNAM, следующим образом:

1) текущая строка имени просматривается в поисках двоеточия. Если в этой строке двоеточие найдено, то делается вывод, что почтовый ящик должен быть расположен в разделяемой памяти и трансляция продолжается по следующей схеме:

- часть текущей строки имени, расположенная справа от двоеточия, помещается в буфер имени-почтового-ящика. Часть

текущей строки имени, расположенная слева от двоеточия, становится новой текущей строкой имени;

- к текущей строке имени добавляется префикс "LNM" и результат подвергается трансляции логического имени;

- если результат содержит логическое имя, то шаги 1 и 2 повторяются до тех пор, пока трансляция не приведет к ошибке, либо пока число трансляций не превысит числа, заданного при генерации системы параметром LNM$C_MAXDEPTH;

- префикс "LNM" удаляется из текущей строки имени, которое уже не необходимо транслировать. Это имя становится именем-разделяемой-памяти. Текущая строка, содержащаяся в буфере имени-почтового-ящика, становится логическим именем с эквивалентным именем MBAN (N - число, назначаемое операционной системой МОС ВП).

2) если почтовый ящик расположен в локальной памяти, то трансляция продолжается по следующей схеме:

- к текущей строке имени добавляется префикс "LNM", а результат подвергается трансляции логического имени;

- если результат является логическим именем, то шаг 1 повторяется до тех пор, пока трансляция не приведет к ошибке, либо пока число трансляций не превысит числа, заданного при генерации системы параметром LNM$C_MAXDEPTH;

- префикс "LNM" удаляется из текущей строки имени, которое уже не требуется транслировать. Эта текущая строка становится логическим именем с эквивалентным именем MBAN:

Примеры 1 и 2 иллюстрируют технику использования логических имен.

Пример 1.

```
¤DEFINE MBX¤CHKPNT        SHRMEM¤1:CHKPNT
```

Пример 2.

```
MBXDESC:        .ASCID /CHKPNT/        дескриптор для логи-
                                       ческого имени почто-
                                       вого ящика
                .
                .
                ¤CREMBX_S LOGNAM=MBXDESC,...
```

Выполняется следующая трансляция логического имени:

1) к CHKPNT добавляется префикс MBX¤;

2) имя MBX¤CHKPNT транслируется в SHRMEM¤1:CHKPNT. Поскольку дальнейшая трансляция не приводит к успеху, то создается логическое имя CHKPNT с эквивалентным именем MBAN: ("N" является параметром, назначенным операционной системой МОС ВП).

Есть два исключения из метода трансляции логических имен, который обсуждается в этом пункте:

1) если строка имени начинается с символа подчеркивания (_), операционная система МОС ВП удаляет этот символ и рассматривает результирующую строку как действительное имя (т.е. Дальнейшая трансляция не выполняется);

2) если строка имени является результатом трансляции логического имени, то строка имени проверяется, имеется ли в ней атрибут TERMINAL. Если строка имени отмечена атрибутом TERMINAL, операционная система МОС ВП рассматривает результирующую строку как действительное имя (т.е. дальней-

шая трансляция не выполняется).

7.13.2. Системные почтовые ящики

Операционная система МОС ЭП использует почтовые ящики для связи между процессами системы. Все сообщения системных почтовых ящиков содержат в первом слове каждого сообщения константу, которая идентифицирует отправителя этого сообщения. Эти константы имеют символические имена (определяемые в макрокоманде ¤MSGDEF) в следующем формате: MSG¤_SENDER.

В табл. 33 содержатся имена, определяемые макрокомандой ¤MSGDEF, и описаны их значения.

Таблица 33

Символические имена	Значение
MSG¤_TRMUNSOLIC	Незатребованные данные терминала
MSG¤_CRUNSOLIC	Незатребованные данные устройства ввода перфокарт
MSG¤_ABORT	Связь по сети с партнером отменена
MSG¤_CONFIRM	Подтверждение связи в сети
MSG¤_CONNECT	Инициирование связи с прибывшим в сеть
MSG¤_DISCON	Зависание при отключении партнера по сети
MSG¤_EXIT	Партнер по сети выбыл преждевременно

Продолжение табл. 33

Символические имена	Значение
MSG¤_INTMSG	Сообщение о прерывании в сети; незатребованные данные
MSG¤_PATHLOST	Маршрут в сети потерян для партнера
MSG¤_PROTOCOL	Ошибка в протоколе сети
MSG¤_REJECT	Отказ от связи в сети
MSG¤_THIRDPARTY	Отсоединение третьего адресата в сети
MSG¤_TIMEOUT	Перерыв связи в сети
MSG¤_NETSHUT	Отключение сети
MSG¤_NODEACC	Узел стал доступным
MSG¤_NODEINACC	Узел стал недоступным
MSG¤_EVTAVL	События доступны программе регистрации событий сети
MSG¤_EVTRCVCHG	Смена базы данных получателя событий
MSG¤_INCDAT	Доступные входные незатребованные данные
MSG¤_RESET	Запрос на сброс виртуального цикла
MSG¤_LINUP	Линия PVC включена
MSG¤_LINDWN	Линия PVC отключена
MSG¤_EVTXMTCHG	Смена базы данных передатчика событий

Остальная часть сообщений содержит переменную информацию, зависящую от системной компоненты, которая послала это

сообщение.

Формат переменной информации для каждого типа сообщений документирован вместе с системной программой, использующей почтовый ящик.

7.13.3. Почтовые ящики для сообщений о завершении процессов

Когда процесс создает другой процесс, он может указать номер устройства почтового ящика в качестве аргумента для программы системного обслуживания ¤CREPRC. Когда созданный процесс удаляется, операционная система МОС ВП посылает сообщение в указанный почтовый ящик завершения.

Почтовые ящики, расположенные в памяти, разделяемой многими процессорами, не могут использоваться в качестве почтовых ящиков завершения.

7.13.4. Почтовые ящики для системных процессов

Следующие программы обслуживания ввода-вывода используются внутри системных процессов для связи с информацией разного рода.

1) послать сообщение контроллеру заданий (¤SNDJBC);

2) послать сообщение оператору (¤SNDOPR).

8. Служебные программы управления процессом

При подключении пользователя операционная система МОС ВЛ создает процесс для выполнения образов. Пользователь может создать другой процесс для выполнения некоторого образа, выдавая команды RUN или SPAWN, используя любые квалификаторы, допустимые для создания процесса. Кроме того, можно написать программу, которая создает новый процесс для выполнения определенного образа.

Для управления процессом используются следующие служебные программы:

1) создать процесс (¤CREPRC);
2) удалить процесс (¤DELPRC);
3) приостановить процесс (¤SUSPND);
4) возобновить процесс (¤RESUME);
5) перевести в "спячку" (¤HIBER);
6) разбудить (¤WAKE);
7) запланировать пробуждение (¤SCHDWK);
8) отменить пробуждение (¤CANWAK);
9) выход (¤EXIT);
10) принудительный выход (¤FORCEX);
11) об'явить обработчик выхода (¤DCLEXH);
12) отменить обработчик выхода (¤CANEXH);
13) установить имя процесса (¤SETPRN);
14) установить приоритет (¤SETPRI);
15) установить привилегии (¤SETPRV);
16) установить режим ожидания ресурса (¤SETRWM);

17) получить информацию о задании или процессе (#GETJPI).

Служебные программы управления процессом позволяют создавать процессы и управлять одним или группой процессов. Данный раздел описывает некоторые аспекты программ управления процессом и включает описание следующих понятий:

1) подпроцессы и отсоединенные процессы;

2) контекст выполнения процесса;

3) создание процесса;

4) межпроцессорное управление и связи;

5) спячка и приостановка процесса;

6) выход образа и обработчики выхода;

7) удаление процесса и сообщения об окончании.

8.1. Подпроцессы и отсоединенные процессы

Каждый процесс является либо подпроцессом, либо отсоединенным процессом. Подпроцесс получает часть квоты ресурсов создавшего его процесса и должен заканчиваться раньше этого процесса. Отсоединенный процесс полностью независим. Например, процесс, который операционная система МОС ЭП создает при подключении пользователя, является отсоединенным процессом.

Системная программа #CREPRC создает как подпроцессы, так и отсоединенные процессы. Количество подпроцессов, которые может создать один процесс, регулируется его квотой PRCLM (максимальное количество процессов). Возможность создания отсоединенного процесса с кодом идентификации пользо-

вателя, отличным от кода идентификации пользователя (UIC) создающего процесса, обеспечивается привилегией DETACH.

8.2. Контекст выполнения процесса

Контекст выполнения процесса определяет процесс в операционной системе МОС ВП. Он включает следующее:

1) образ, который выполняет данный процесс;

2) входные и выходные потоки для образа, работающего в процессе;

3) принимаемые по умолчанию диск и каталог для данного процесса;

4) квоты системных ресурсов и привилегии пользователя, доступные данному процессу.

Когда операционная система МОС ВП создает отсоединенный процесс как результат подключения пользователя, она использует для определения контекста выполнения процесса системный файл авторизации пользователя (SYSUAF.DAT).

Когда пользователь подключается к операционной системе МОС ВП, происходит следующее:

1) процесс, созданный для пользователя, выполняет образ с именем LOGINOUT;

2) терминал, который использует пользователь, определяется как устройство входного и выходного потока и потока сообщений об ошибках для образов, выполняемых данным процессом;

3) имена по умолчанию для дисков и каталогов пользователя берутся из файла авторизации пользователя;

4) с созданным процессом связываются квоты ресурсов и привилегии, предоставленные пользователю системным администратором;

5) в созданный процесс отображается интерпретатор языка команд.

Если для создания процесса вызывается системная программа $CREPRC, то контекст определяется при помощи задания аргументов программы.

8.3. Создание процесса

В пп. 8.3.1...8.3.5. приведены примеры создания процесса и описывается, как аргументы системной программы $CREPRC определяют контекст процесса.

8.3.1. Определение выполняемого образа для подпроцесса

При вызове программы системного обслуживания $CREPRC следует указывать аргумент IMAGE, чтобы обеспечить процессу имя образа, который должен выполняться. Пример иллюстрирует создание процесса для выполнения образа с именем CARRIE.EXE.

Пример.

```
PROGNAME:
        .ASCID  /CARRIE/ ; дескриптор для образа, который
                           должен выполняться
        $CREPRC_S -       создать процесс для выполнения
```

CARRIE

IMAGE=PROGNAME

В приведенном примере указано только имя файла; служебная программа использует текущие значения по умолчанию для дисков и каталогов, осуществляет трансляцию логических имен, по умолчанию принимает тип файла - EXE и находит наиболее позднюю версию файла образа. Подпроцесс удаляется после того, как он завершит выполнение образа.

8.3.2. Входные, выходные и диагностические устройства для подпроцессов

При вызове программы системного обслуживания $CREPRC можно указать эквивалентные имена для логических имен SYS$INPUT, SYS$OUTPUT и SYS$ERROR. Такие пары "логическое имя - эквивалентное имя" помещаются в таблицу логических имен для создаваемого процесса.

В примере 1 представлен фрагмент программы определения входных, выходных и диагностических устройств для подпроцесса. В примечаниях описано, как используются эти устройства.

Пример 1.

```
INSTREAM: -
        .ASCID  /SUB_MAIL_BOX/      дескриптор для входного
                                    потока
OUTSTREAM:
        .ASCID  /COMPUTE_OUT/       дескриптор для выходного
                                    и диагностического пото-
```

)

```
                                    ка
PROGNAME: -                         дескриптор для имени об-
                                    раза
        .ASCID  /COMPUTE.EXE/


        .
        ¤CREPRC_S -                 создать процесс
                IMAGE=PROGNAME,-
                INRUT=INSTREAM,-
                OUTPUT=OUTSTREAM,-
                ERROR=OUTSTREAM
```

Примечания:

1. Аргумент INPUT отождествляет эквивалентное имя SJB_MAIL_BOX с логическим именем SYS¤INPUT. Это логическое имя может представлять почтовый ящик, который вызывающий процесс предварительно создал при помощи программы системного обслуживания ¤CREMBX. Все данные, которые подпроцесс считывает с логического устройства SYS¤INPUT, будут прочитаны из почтового ящика.

2. Аргумент OUTPUT отождествляет эквивалентное имя COMPUTE_OUT с логическим именем SYS¤OUTPUT. Все сообщения, которые программа пишет на логическое устройство SYS¤OUTPUT, будут записаны в данный файл.

3. Аргумент ERROR отождествляет эквивалентное имя COMPUTE_OUT с логическим именем SYS¤ERROR. Все выдаваемые операционной системой МОС ВП сообщения об ошибках будут

записываться в этот файл. Поскольку тот же самый файл используется для выходных сообщений программы, он фактически содержит всю выходную информацию, полученную во время выполнения образа программы.

Программа системного обслуживания ¤CREPRC не обеспечивает принимаемых по умолчанию эквивалентных имен для логических имен SYS¤INPUT, SYS¤OUTPUT и SYS¤ERROR. Если ничего не указано, то в качестве эквивалентных имен могут быть использованы элементы, если таковые имеются, в групповых или системных таблицах логических имен. Если во время выполнения подпроцесс считывает или записывает на такие логические устройства, для которых не существует эквивалентного имени, то возникает ошибка.

В программе, которая создает подпроцесс, можно сделать так, чтобы данный подпроцесс разделял с создающим процессом входное, выходное или диагностическое устройство. Для этого требуется проделать следующее:

1) получить имя устройства для логического имени SYS¤INPUT, SYS¤OUTPUT или SYS¤ERROR при помощи программы системного обслуживания ¤GETDVI;

2) использовать адрес данного дескриптора при задании аргументов INPUT, OUTPUT или ERROR программы системного обслуживания ¤CREPRC.

Пример 2 иллюстрирует эти действия.

Пример 2.

```
¤PRCDEF
¤DVIDEF
```

00152-01 97 06

```
DVILIST:                                начало списка элемен-
                                        тов программы ¤GETDVI
        .WORD   64                      длиной максимум 16 бай-
                                        тов
        .WORD   DVI¤_DEVNAM             получить имя терминала
        .ADDRESS -                      пункт назначения имени
                TERM                    терминала
        .ADDRESS -                      пункт назначения длины
                TERMDESC                строки
        .LONG           0               конец списка элементов

TERMDESC:                               дескриптор для имени
                                        терминала
        .WORD   64                      длиной максимум 16 бай-
                                        тов
        .WORD   0
TERMADDR:
        .ADDRESS -
                TERM
TERM:   .BLKB   64                      здесь размещается имя
                                        терминала

LOGNAM: .ASCID  /SYS=INPUT/

IMAGENAME:
        .ASCID  /WRKD¤:[ORANGE]MIRROR/      образ для
                                          ; подпроцесса
```

```
        $GETDVI_S -
                DEVNAM=LOGNAM, -    поместить возвращаемую
                                    информацию на SYS$INPUT
                ITMLST=DVILIST      адрес списка элементов

        BLBC    R0,SSERR            в случае неуспешного за-
                                    вершения перейти на про-
                                    грамму обработки ошибки

10$:    $CREPRC_S -                 создать подпроцесс,
                IMAGE=IMAGENAME,-   ; выполняющий образ
                INRUT=TERMDESC,-      MIRROR и использую-
                OUTPUT=TERMDESC,-     щий терминал созда-
                ERROR=TERMDESC,-      ющего процесса в ка-
                                      честве входного, вы-
                                      ходного и диагности-
                                      ческого устройства
                BASPRI=#4           установить базовый при-
                                    оритет, равный 4
```

При работе подпроцесса логические имена SYS$INPUT, SYS$OUTPUT и SYS$ERROR отождествляются с именем логического входного устройства создающего процесса. Кроме того, подп-

роцесс может выбирать одну из следующих возможностей:

1) использовать систему управления данными СУД-32, чтобы открыть файл для чтения и/или записи;

2) использовать программу системного обслуживания ¤ASSIGN для того, чтобы назначить устройству канал ввода-вывода для операций ввода-вывода.

В примере 3 программа назначает канал для устройства, определенного именем SYS¤OUTPUT.

Пример 3.

```
OUTPUT: .ASCID  /SYS¤OUTPUT/        дескриптор логичес-
                                    кого имени
OUTCHAN:
        .BLKW   1                   номер канала выход-
                                    ного устройтсва


        ¤ASSIGN_S -
                DEVNAM=OUTPUT,-
                CHAN=OUTCHAN
```

8.3.3. Принимаемые по умолчанию диск и каталог для создаваемых процессов

Если для создания процесса, выполняющего некоторый образ, используется программа системного обслуживания ¤CREPRC, то операционная система МОС ВП находит файл образа на устройстве и в каталоге, принятых по умолчанию для соз-

даваемого процесса. Каждый создаваемый процесс приобретает текущие значения по умолчанию для устройства и каталога создавшего процесса.

Если созданный процесс выполняет образ, которого нет в его принятом по умолчанию каталоге, то пользователь должен идентифицировать каталог и, если требуется, устройство в спецификации файла этого образа, который должен выполняться.

При вызове программы ¤CREPRC нельзя определить принимаемые по умолчанию устройство и/или каталог для создаваемого процесса, отличные от принимаемых по умолчанию устройства и каталога создающего процесса. Однако, создаваемый процесс может определить эквивалентное имя для логического устройства SYS¤DISK при помощи вызова программы системного обслуживания ¤CRELNM.

Если процесс является подпроцессом, то можно определить эквивалентное имя в групповой таблице логических имен, в таблице логических имен задания или в любой таблице логических имен, разделяемой между создающим процессом и подпроцессом. Созданный процесс может также установить свой собственный принимаемый по умолчанию каталог при помощи вызова програмы управления значениями по умолчанию каталога SYS¤SEtDIR системы СУД-32.

Процесс может создать другой процесс со значением по умолчанию для каталога, отличным от его собственного, следующим образом:

1) процесс, создающий новый процесс, вызывает програм-

му SYS$SETDIR, чтобы изменить свой собственный, принятый по умолчанию каталог;

2) создающий процесс вызывает программу $CREPRC для создания нового процесса;

3) создающий процесс вызывает программу SYS$SETDIR, чтобы изменить свое значение по умолчанию для каталога на то, которое было до первого вызова SYS$SETDIR.

Теперь создающий процесс имеет свой первоначальный, принятый по умолчанию каталог. Новый процесс имеет, принятый по умолчанию каталог отличный от того, который имел создающий процесс при создании нового процесса.

8.3.4. Управление ресурсами создаваемых процессов

Обычно при создании подпроцесса требуется только назначить ему образ, который необходимо выполнять, и, по желанию, устройства SYS$INPUT, SYS$OUTPUT и SYS$ERROR. Операционная система МОС ВП обеспечивает принимаемые по умолчанию значения для привилегий процесса, квот ресурсов, режимов выполнения и приоритета. Однако, в некоторых случаях можно конкретно определить эти значения. Аргументы программы системного обслуживания $CREPRC, которые управляют такими характеристиками следующие:

PRVADR - адрес списка привилегий. Данный аргумент определяет список привилегий для создаваемого процесса. Если этот аргумент не задан, то используются привилегии вызывающего процесса. Если аргумент PRVADR задан, то

используются только привилегии, указанные в битовой маске, привилегии вызывающего процесса не используются. Например, создающий процесс имеет пользовательские привилегии GROUP и TMPMBX. Он создает процесс, указывая пользовательскую привилегию TMPMBX. Созданный процесс получит только пользовательскую привилегию TMPMBX; он не будет иметь пользовательской привилегии GROUP.

Если требуется создать процесс, имеющий привилегию, которой нет у создающего процесса, пользователь должен иметь привилегию SETPRV.

Символические имена, соответствующие привилегиям, определяются макрокомандой $PRVDEF. Каждое символическое имя начинается с PRV$V_ и определяет номер бита, который для задания данной привилегии должен устанавливаться в 1. Пример показывает определение данных для маски, задающей привилегии GRPNAM и GROUP.

Пример.

```
PRVMSK: .LONG   <1@PRV$V_GRPNAM>!<1@PRV$V_GROUP>   имя
                                                   группы и группа
        .LONG   0                                  требуется маска длиной в
                                                   квадрослово. Для данных
                                                   привилегий никакие биты
                                                   в старшем длинном слове
                                                   не устанавливаются
```

QUOTA (квота) - определяет список квот для подпроцесса. Поскольку подпроцесс получает часть квот своего создате-

ля для элементов очереди таймера, буферов ввода-вывода и т.п., то можно управлять размером каждой квоты, назначаемой подпроцессу. Если данный аргумент не задан, то операционная система МОС ВП определяет для подпроцесса принимаемые по умолчанию квоты. Однако, если создающий процесс сам имеет только принимаемые по умолчанию квоты, требуется задавать этот аргумент, чтобы предотвратить захват подпроцессом всех отбираемых у процесса квот;

STSFLG (флаг состояния) - определяет флаг состояния, т.е. набор битов, которые управляют некоторыми характеристиками выполнения созданного процесса, включая режим ожидания ресурса и режим обмена процесса;

BASPRI (базовый приоритет) - устанавливает базовый приоритет выполнения для созданного процесса. Если он не задан, то по умолчанию устанавливается приоритет 2 для языков макроассемблер и блисс и 0 для других языков. Если желательно, чтобы у подпроцесса был более высокий приоритет, чем у его создателя, пользователь должен иметь привилегию ALTPRI, позволяющую повысить уровень приоритета.

8.3.5. Отсоединенные процессы

Создание отсоединенного процесса - это, в первую очередь, функция, выполняемая операционной системой МОС ВП во время подключения пользователя. Привилегия DETACH управляет возможностью создавать отсоединенный процесс с UIC, отлич-

ным от UIC создающего процесса. Единственной возможностью для определения, будет ли процесс подпроцессом или отсоединенным процессом, является аргумент UIC программы системного обслуживания $CREPRC; он обеспечивает код идентификации пользователя (UIC) для созданного процесса. Если аргумент UIC пропущен, то программа системного обслуживания $CREPRC создает подпроцесс, который работает под кодом идентификации пользователя создающего процесса.

Можно также создать отсоединенный процесс с тем же UIC, который имеет создающий процесс, задавая флаг в аргументе STSFLG. Для того, чтобы создать отсоединенный процесс с таким же кодом идентификации пользователя, как и у создающего процесса, привилегии DETACH не требуется.

8.4. Межпроцессорные связи и управление

Процессы могут быть либо полностью независимыми, либо взаимосвязанными. Вопросы, обсуждаемые в пп. 8.4.1 и 8.4.2. касаются областей применения, которые требуют единовременного выполнения многих программ.

8.4.1. Ограничения на создание и управление процессом

Существует три уровня привилегий управления процессом:

1) процессы с одинаковым кодом идентификации пользователя (UIC) всегда могут выполнять служебные программы управления процессом друг для друга;

2) для того, чтобы процессы могли выполнять служебные программы управления процессом для других процессов, работающих в этой же группе, требуется привилегия GROUP;

3) для того, чтобы процесс мог выполнять служебные программы управления процессом для всех процессов в системе, он должен иметь привилегию WORLD.

Для выполнения некоторых специальных функций, например, для установления процессу базового приоритета более высокого уровня, чем тот, который имеет запрашивающий процесс, требуются дополнительные привилегии.

8.4.2. Идентификация процесса

Существует два типа идентификации процесса:

1) номер идентификации процесса (PID). Когда создается процесс, операционная система МОС ЭП присваивает ему уникальный 32-битовый номер. Если пользователь обеспечил для программы системного обслуживания $CREPRC аргумент PIDADR, то операционная система МОС ЭП возвращает номер идентификации процесса в указанное место. Потом пользователь может использовать этот номер идентификации процесса в последующих служебных программах управления процессом;

2) имя процесса. Именем процесса является символьная строка длиной от 1 до 15 символов. Каждое имя процесса должно быть уникальным в пределах одной и той же группы (процессы, принадлежащие разным группам, могут иметь одинаковые имена). Имя процессу можно присвоить при его создании, указывая аргумент PRCNAM. Это имя затем можно исполь-

зовать для ссылок на данный процесс в вызовах других программ системного обслуживания. Нельзя использовать имя процесса для идентификации процесса, не принадлежащего к той же группе, что и вызывающий процесс; в этом случае необходимо использовать номер идентификации процесса.

Если требуется управлять выполнением подпроцесса, необходимо дать ему имя. Кроме того, следует давать имена отсоединенным процессам, работающим в одной и той же группе, если они связаны друг с другом или выполняют управляющие функции, оказывающие влияние друг на друга.

Пример 1.

```
ORION:  .ASCID  /ORION/          дескриптор для
                                 имени процесса
ORIONID:
        .LONG   0                возвращаемый номер
                                 идентификации процесса


        $CREPRC_S -
                PRCNAM=ORION,-
                PIDADR=ORIONID,...
```

Служебная программа возвращает номер идентификации в длинное слово ORIONID. Для ссылок на данный процесс в других вызовах программ системного обслуживания можно использовать либо имя процесса (ORION), либо номер идентификации процесса (ORIONID).

Процесс может сам установить собственное имя или изме-

нить его при помощи программы системного обслуживания ¤SETPRN.

Пример 2.

```
CYGNUS: .ASCID  /CYGNUS/        дескриптор для имени
                                процесса

        ¤SETPRN_S -
                PRCNAM=CYGNUS
```

Большинство служебных программ управления процессом допускает либо аргумент PRCNAM, либо PIDADR, или оба эти аргумента. Однако, лучше идентифицировать процесс номером идентификации процесса по следующим причинам:

1) служебная программа выполняется быстрее, потому что не требуется искать имена процессов в таблице;

2) номер идентификации процесса необходим для процессов, выполняющихся в другой группе (см. подпункт 8.4.2.1.).

Если не указаны ни аргумент PRCNAM, ни аргумент PIDADR, то служебная программа выполняется для вызывающего процесса. Возможные комбинации этих аргументов и пояснения, как служебные программы интерпретируют их, приведены в табл. 34.

Таблица 34

Идентификация процесса

Имя процесса задано?	Адрес номера идентификации процесса задан?	Номер идентификации процесса содержит:	Действия служебных программ
Нет	Нет	-	Используется идентификация вызывающего процесса. Номер идентификации процесса не возвращается
Нет	Да	0	Используется и возвращается номер идентификации вызывающего процесса
Да	Нет	-	Используется имя процесса. Номер идентификации процесса не возвращается
Да	Да	0	Используется имя процесса и возвращается номер идентификации процесса
Да	Да	Номер идентификации	Используется и возвращается номер иден-

Продолжение табл. 34

Имя процесса задано?	Адрес номера идентификации процесса задан?	Номер идентификации процесса содержит:	Действия служебных программ
		процесса	тификации процесса; имя процесса игнорируется

3.4.2.1. Присвоение имен процессам внутри групп

В квалификационные имена процессов всегда добавляется номер их группы. Операционная система МОС ВП строит таблицу всех имен процессов и кодов идентификации пользователя (UIC), связанных с каждым именем процесса. Если в служебной программе управления процессом используется аргумент PRCNAM, то в таблице ищется элемент, который содержит заданное имя процесса и номер группы вызывающего процесса.

Вызывающий процесс должен иметь привилегию GROUP для того, чтобы использовать служебные программы управления процессом для процессов внутри своей группы; эта привилегия не требуется, если указан процесс с тем же кодом идентификации пользователя (UIC), который имеет вызывающий процесс.

Если заданное имя процесса не имеет тот же номер группы, что и вызывающий процесс, то поиск имени процесса заканчивается по ошибке. Поиск заканчивается по ошибке даже

в том случае, если вызывающий процесс имеет пользовательскую привилегию WORLD. Для того, чтобы выполнить служебную программу управления процессом для процесса из другой группы, запрашивающий процесс должен использовать номер идентификации процесса и иметь пользовательскую привилегию WORLD.

8.4.2.2. Получение информации о процессах

Программа системного обслуживания $GETJPI позволяет процессу получить информацию о нем самом или о другом процессе.

8.4.2.3. Средства межпроцессной связи

Для связи процессов существуют следующие средства:

1) кластер общих флагов событий;

2) логические имена;

3) почтовые ящики;

4) глобальные секции;

5) программы системного обслуживания управления захватом;

6) файлы.

Каждое средство связи предоставляет различные возможности в смысле скорости передачи информации и количества передаваемой информации. Например, файлы дают возможность разделять практически неограниченное количество информации; в то же время использование файлов - это самое медленное средство, потому что для получения информации должен быть

осуществлен доступ к диску.

Аналогично файлам глобальные секции позволяют разделять большие об'емы информации. Так как разделение информации через глобальные секции требует только доступа к памяти, оно является самым быстрым средством связи.

Логические имена и почтовые ящики позволяют передавать средние об'емы информации. Поскольку оба эти средства работают через относительно сложные программы системного обслуживания, они осуществляют передачу быстрее, чем файлы, но медленнее, чем остальные средства связи.

Служебные программы управления захватом и кластер общих флагов событий позволяют передавать небольшие об'емы информации. Это самые быстрые, за исключением глобальных секций, средства межпроцессной связи.

Кластеры общих флагов событий

Процессы, работающие в пределах одной и той же группы, могут использовать кластеры общих флагов событий для того, чтобы сигнализировать о возникновении или завершении отдельных действий (см. раздел 4).

Таблицы логических имен

Процессы, работающие в одном и том же задании, могут получать эквивалентные имена для логических имен таблицы логических имен задания. Процессы, работающие в одной и той же группе, могут использовать групповую таблицу логических имен. Для того, чтобы процесс мог помещать имена в группо-

вую таблицу логических имен, он должен иметь пользовательскую привилегию GRPNAM. Системную таблицу логических имен могут использовать все процессы в системе. Кроме того, процессы могут создавать и использовать определяемые пользователем таблицы логических имен (см. раздел 6).

Почтовые ящики

Почтовые ящики можно использовать как виртуальные устройства ввода-вывода для передачи между процессами информации, сообщений и данных (см. раздел 7). Кроме того, почтовые ящики можно использовать для того, чтобы дать возможность создающему процессу определить, когда и при каких условиях был удален созданный им подпроцесс.

Глобальные секции

Глобальными секциями могут быть либо файлы на дисках, либо секции страничного файла, содержащие разделяемые коды или данные. При помощи служебных программ управления памятью эти файлы могут быть отображены на виртуальное адресное пространство нескольких процессов. В том случае, если глобальная секция представляет собой файл данных на диске, взаимосвязанные процессы могут синхронизировать чтение и запись данных в физической памяти; как только данные обновляются, системное средство обмена страниц перезаписывает обновленные данные непосредственно в файл на диске. Глобальные секции страничного файла целесообразно использовать для временного хранения общих данных; они не отобра-

жаются в файл на диске, это просто страницы из принятого по умолчанию системного страничного файла (см. Подраздел 11.6).

Программы системного обслуживания управления захватом

Процессы могут использовать программы системного обслуживания управления захватом для того, чтобы управлять доступом к ресурсам (т.е. к любому об'екту в операционной системе МОС ВП, который данный процесс может выполнить, или читать с него, или записывать на него).

Помимо управления доступом, программы системного обслуживания управления захватом обеспечивают механизм для передачи информации между процессами, имеющими доступ к ресурсу (блоки занчений захвата). Для того, чтобы сообщить процессу, что другие процессы ждут ресурс, можно использовать асинхронные системные прерывания (AST) по захвату (см. разлел 12).

8.5. Спячка и приостанов процесса

Существуют два способа временно прекратить выполнение процесса:

1) спячка, осуществляемая программой системного обслуживания $HIBER;

2) приостановка, осуществляемая программой системного обслуживания $SUSPND.

Процесс может нормально продолжить выполнение только после вызова соответствующей программы системного обслужи-

вания ¤WAKE, если он был в спячке, или после вызова программы системного обслуживания ¤RESUME, если он был приостановлен.

Сравнение состояний спячки и приостановки процесса приведено в табл. 35.

Таблица 35

Спячка и приостановка процесса

Спячка	Приостановка
Можно вызвать спячку только для самого себя	Можно приостанавливать себя или другой процесс в зависимости от привилегии
Возвращается в исходное состояние программой системного обслуживания ¤WAKE	Возвращается в исходное состояние программой системного обслуживания ¤RESUME
Прерываемое состояние; может получать асинхронные системные прерывания AST	Непрерываемое состояние; не может получать асинхронные системные прерывания AST
Может пробудить себя	Не может возобновить себя
Возможно пробуждение в определенный момент времени или через фиксированные интервалы времени	Не может возобновиться по времени
Требует мало системных накладных расходов	Требует системной динамической памяти

8.5.1. Спячка процесса

Механизм "спячка/пробуждение" обеспечивает эффективное средство, чтобы подготовить образ для выполнения и затем перевести его в состояние ожидания, пока он не потребуется. Когда выдается запрос на пробуждение, образ снова активизируется с малой задержкой.

Например, если создается подпроцесс, который будет выполнять повторно одну и ту же функцию и должен работать сразу же, как только это потребуется, то можно использовать программы системного обслуживания $HIBER и $WAKE, как показано в примере.

Вариантом программы системного обслуживания $WAKE является пробуждение по времени - пробуждение процесса, находящегося в спячке, в фиксированный момент времени или по истечении интервала времени. Эту функцию осуществляет программа системного обслуживания $SCHDWK. При помощи программы системного обслуживания $SCHDWK процесс может задавать пробуждение по времени для самого себя перед тем, как вызвать программу $HIBER.

Процессы, находящиеся в спячке, могут быть прерваны асинхронными системными прерываниями (AST) при условии если разрешена доставка AST. Процесс может сделать для себя вызов программы $WAKE в процедуре обслуживания AST и продолжить работу после выполнения процедуры обслуживания AST (см. раздел 5).

Пример.

```
PROCESS TAURUS                  ; процесс TAURUS
ORION:  .ASCID  /ORION/           дескриптор для имени
                                  подпроцесса
FASTCOMP:
        .ASCID  /COMPUTE.EXE/     дескриптор для имени
                                  образа

¤CREPRC_S -                       создать процесс ORION
                PRCNAM=ORION,-
                IMAGE=FASTCOMP,...
        BSBW    ERROR             продолжить

¤WAKE_S PRCNAM=ORION              пробудить от спячки ORION
        BSBW    ERROR

        ¤WAKE_S PRCNAM=ORION      снова пробудить от спячки
                                  ORION
        BSBW    ERROR

PROCESS ORION                     процесс ORION
        .ENTRY  COMPUTE,^M<>      входная маска
10¤:    ¤HIBER_S                  спячка
```

00152-01 97 06

```
BSBW    ERROR                   выполнение

BRW     10¤                     возврат к спячке
```

Примечания:

1. Процесс TAURUS создает процесс ORION, задавая дескриптор для образа с именем COMPUTE.

2. Инициируется образ COMPUTE, и процесс ORION вызывает программу системного обслуживания ¤HIBER.

3. В соответствующий момент времени процесс TAURUS вызывает программу ¤WAKE для процесса ORION. ORION продолжает выполнение действий, следующих после вызова служебной программы ¤HIBER. По завершении своей работы ORION возвращается в начало цикла, повторяет вызов ¤HIBER и ждет следующего пробуждения от спячки.

8.5.2. Альтернативные средства спячки

Можно использовать два дополнительных способа перевода процесса в состояние спячки:

1) задать аргумент STSFLG для программы системного обслуживания ¤CREPRC, установив тот бит, который заставляет программу ¤CREPRC перевести создаваемый процесс в состояние спячки сразу после инициализации процесса;

2) задать квалификаторы /DELAY, /SCHEDULE или /INTERVAL в команде RUN.

Если используется первый способ, то создающий процесс может управлять моментом пробуждения созданного процесса.

если используется команда RUN, то моментом пробуждения процесса управляют квалификаторы.

Если задан квалификатор /INTERVAL и образ, который должен выполняться, не содержит вызова программы системного обслуживания ¤HIBER, то этот образ переводится в состояние спячки всякий раз, когда будет выдана инструкция RET. Каждый раз, когда образ пробуждается от спячки, он начинает выполнение с его точки входа. Если образ содержит вызов программы ¤HIBER, то при каждом пробуждении от спячки он начинает выполнение либо с точки, следующей за вызовом программы ¤HIBER, либо с точки входа (если последней была выдана инструкция RET).

Если запросы на пробуждение выполняются по истечении интервалов времени, то образ может заканчиваться программами системного обслуживания ¤DELPRC или ¤FORCEX, или на уровне команд языка DCL - командой STOP.

Данные средства позволяют писать программы, которые могут выполняться однократно, по запросу или в цикле. Такая программа должна обеспечивать не только целостность областей данных, которые модифицируются во время ее выполнения, но и состояния открытых файлов.

8.5.3. Приостановка

При помощи программы системного обслуживания ¤SUSPND процесс может перевести себя или другой процесс в состояние ожидания, подобного спячке. Операционная система МОС ВП не обеспечивает системных средств для принудительной выгрузки

процесса, но программа системного обслуживания ¤SUSPND позволяет решить эту задачу. Приостановленные процессы выбираются для выгрузки в первую очередь. Приостановленный процесс нельзя прервать асинхронными системными прерываниями AST, и он может возобновить выполнение только после того, как другой процесс выдает для него запрос на программу системного обслуживания ¤RESUME. Если асинхронные системные прерывания AST были поставлены в очередь к приостановленному процессу, то они будут доставлены, когда данный процесс возобновит выполнение. Это является эффективным средством блокирования доставки всех AST.

8.6. Завершение выполнения образа

Если выполнение образа завершается нормально, то операционная система МОС ЭП выполняет ряд функций по свертыванию образа. Если образ выполняется командным интерпретатором, то свертывание образа подготавливает процесс для выполнения другого образа. Если образ выполнялся не командным интерпретатором, а, например, подпроцессом, то этот подпроцесс удаляется.

Такие завершающие действия инициируются также в том случае, если образ завершился аварийно в результате одного из следующих условий:

1) условия ошибки спецификаций, вызванные ошибочными спецификациями при создании процесса. Например, если задано неверное имя для логического имени SYS¤INPUT, SYS¤OUTPUT, SYS¤ERROR или указано неверное или несуществующее имя обра-

за, то в создаваемый процесс передается сигнал о состоянии ошибки;

2) при возникновении исключительной ситуации управление передается какому-нибудь определенному пользователем обработчику кодов состояния для обработки данной исключительной ситуации. Если определенных пользователем обработчиков кодов состояния нет, то управление получает об'явленный операционной системой МОС ВП обработчик кодов состояния, который инициирует действия, завершающие образ. (см. раздел 10).

3) программа системного обслуживания $FORCEX вызвана для данного процесса другим процессом.

8.6.1. Действия по свертыванию образа

Операционная система МОС ВП выполняет функции свертывания образа, которые освобождают системные ресурсы, полученные процессом во время его выполнения в пользовательском режиме. Эти действия и порядок, в котором они выполняются, следующие:

1) отменяются все выставленные запросы ввода-вывода на каналы ввода-вывода и каналы ввода-вывода освобождаются;

2) удаляются страницы памяти, занятые образом или назначенные ему, и размер рабочего набора процесса снова принимает значение, принятое по умолчанию;

3) освобождаются все устройства, назначенные данному процессу в пользовательском режиме (устройства, назначенные из потока команд языка DCL в режиме супервизора, не осво-

бождаются);

4) отменяются все запросы, планируемые по таймеру, включая запросы на пробуждение;

5) отсоединяются кластеры общих флагов событий;

6) асинхронные системные прерывания пользовательского режима, которые поставлены в очередь, но не были доставлены, отменяются;

7) отменяются векторы исключительных ситуаций, об'явленные в пользовательском режиме, обработчики режима совместимости и обработчики изменения режима в пользовательский режим;

8) запрещается режим исключительной ситуации по ошибке программ системного обслуживания;

9) удаляются все личные логические имена процесса и таблицы логических имен, созданные для пользовательского режима. Удаление таблицы логических имен приводит к удалению всех имен в этой таблице. Имена, входящие в разделяемые таблицы логических имен, такие как групповая таблица или таблица задания, не удаляются при свертывании образа независимо от режима доступа, для которого они создавались.

8.6.2. Программа системного обслуживания $EXIT

Для того, что инициировать действия по свертыванию образа, операционная система МОС ВП от имени данного процесса вызывает программу системного обслуживания $EXIT. В некоторых случаях процесс может сам вызвать программу $EXIT для завершения образа, например, если возникла неопознава-

мая ошибка. Программа системного обслуживания ¤EXIT принимает в качестве аргумента код состояния. Если программа ¤EXIT применяется для прекращения выполнения образа, то можно использовать этот аргумент кода состояния для того, чтобы передать информацию о завершении образа. Если возврат из образа происходит без вызова программы ¤EXIT, то текущее значение в регистре R0 передается как код состояния при вызове системной программы ¤EXIT.

Код состояния используется следующим образом:

1) командный интерпретатор, если он получает управление после свертывания образа, использует код состояния для выдачи на экран возможного сообщения об ошибке;

2) если образ об'явил обработчик выхода, то код состояния записывается по адресу, указанному в блоке управления выходом;

3) если процесс был создан другим процессом, и создавший процесс указал почтовый ящик для получения информации об окончании, то код состояния, когда данный процесс удаляется, записывается в указанный почтовый ящик окончания.

8.6.3. Обработчики выхода

Обработчики выхода - это программы, которые могут выполнять специальную очистку образа или операции по свертыванию. Например, если образ использует память для буферизованных данных, то обработчик выхода может обеспечить, чтобы эти данные не потерялись, если завершение работы образа происходит в результате возникновения ситуации ошиб-

ки.

Чтобы создать процедуру управления выходом, необходимо организовать блок управления выходом и указать адрес этого блока управления в вызове программы системного обслуживания $DCLEXH. Обработчики выхода вызываются при помощи стандартных соглашений по вызову процедур. Аргументы для обработчика выхода можно обеспечить в блоке управления выходом. Первый аргумент в списке аргументов блока управления выходом должен задавать адрес длинного слова, в которое операционная система МОС ВП записывает код состояния из программы $EXIT.

Если образ об'являет несколько обработчиков выхода, то блоки управления выходом связываются вместе по принципу "последним пришел - первым ушел". После того, как обработчик был вызван и вернул управление, данный блок управления удаляется из списка. Кроме того, блоки управления выходом можно удалять перед выходом образа при помощи программы системного обслуживания $CANEXH.

Обработчики выхода могут быть об'явлены из системных программ, работающих в режиме супервизора или в режиме управления. Блоки управления выходом таких обработчиков выхода также связываются вместе в другие списки и получают управление после того, как будут выполнены обработчики выхода, об'явленные из пользовательского режима.

Обработчики выхода вызываются как часть программы системного обслуживания $EXIT. Не существует гарантии того, что обработчики выхода будут вызваны в том случае, если

образ заканчивается нестандартно, например, по вызову программы ¤DELPRC из другого процесса.

8.6.4. Принудительный выход

Программа системного обслуживания ¤FORCEX позволяет процессу инициировать свертывание образа для другого процесса.

Пример 1.

```
CYGNUS: .ASCID  /CYGNUS/        дескриптор имени
                                процесса


        ¤FORCEX_S -
                PRCNAM=CYGNUS
```

Поскольку программа системного обслуживания ¤FORCEX вызывает программу системного обслуживания ¤EXIT, то перед свертыванием образа выполняются все обработчики выхода, об'явленные для данного образа. Если процесс использует командный интерпретатор, то этот процесс не отменяется и может выполнять другой образ. Так как программа системного обслуживания ¤FORCEX использует механизм асинхронных системных прерываний AST, то выход нельзя выполнить в том случае, если процесс, для которого осуществляется принудительный выход, запретил доставку асинхронных системных прерываний. (см. раздел 5).

Пример 2 иллюстрирует фрагмент программы, показывающий пример процедуры обработки выхода.

Пример 2.

```
EXITBLOCK:                  блок управления выходом
        .LONG   0           система использует это как
                            указатель
        .ADDRESS -          адрес обработчика выхода
                EXITRTN
        .LONG   1           количество аргументов
        .ADDRESS -          адрес длинного слова, при-
                STATUS      нимающего код состояния
STATUS: .BLKL   1           код состояния из программы
                            $EXIT


        .ENTRY  PEGASUS,^M<R2,R3>     маска входа для
                                      PEGASUS
        $DCLEXH_S -
                        DESBLK=EXITBLOCK   ; об'явить
                                   обработчик выхода
        BSBW    ERROR


        RET                 конец основной программы

        .ENTRY  EXITRTN,^M<R2>      маска входа
        BLBS    STATUS,10$          нормальный выход?
```

да, закончить

нет, очистка

10#: RET закончить

Примечания:

1. EXITBLOCK - это блок управления выходом для обработчика выхода EXITRTN. Третье длинное слово определяет количество передаваемых аргументов. В данном примере передается только один аргумент - адрес длинного слова, в которое операционная система МОС ЭП запишет возвращаемый код состояния. Данный аргумент должен обеспечиваться в блоке управления выходом.

2. Вызов программы системного обслуживания #DCLEXH определяет адрес блока управления выходом, тем самым об'являя EXITRTN как обработчик выхода.

3. EXITRTN проверяет код состояния. В случае нормального выхода EXITRTN возвращает управление. В противном случае обрабатывает условие ошибки.

8.7. Удаление процесса

Удаление процесса приводит к полному уничтожению процесса в операционной системе МОС ЭП. Процесс может удаляться при одном из следующих событий:

1) вызвана программа системного обслуживания #DELPRC;

2) удален процесс, который создал данный подпроцесс;

3) интерактивный процесс использует команду LOGOUT диалогового командного языка DCL;

4) пакетное задание достигает конца своего командного файла;

5) интерактивный процесс использует команду диалогового командного языка "STOP/ID = номер идентификации процесса" или "STOP имя пользователя";

6) процесс, который содержит единственный образ, вызывает программу системного обслуживания $EXIT.

Процесс нельзя удалить, пока не будут удалены все подпроцессы, которые он создал. Когда операционная система МОС ЭП получает указание для удаления процесса в результате какого-либо из перечисленных условий, она сначала находит все подпроцессы, ведя поиск по иерархии. Низший по иерархии подпроцесс - это тот, который не имеет собственных подпроцессов-потомков. Когда такой подпроцесс удален, породивший его подпроцесс становится подпроцессом, который не имеет подпроцессов-потомков, и его можно удалить. Верхний в иерархии процесс - это процесс, который в конечном счете является порождающим процессом для всех других подпроцессов.

Для всех процессов, начиная с низшего в иерархии и заканчивая высшим, выполняются следующие действия:

). свертывается образ, работающий в данном процессе. (см. п. 8.6.1). При удалении процесса программа свертывания образа освобождает все системные ресурсы, включая те, которые были получены из режимов доступа, отличных от пользова-

тельского режима;

). если удаляемый процесс является подпроцессом, то квоты ресурсов отдаются создавшему процессу;

). если создающий процесс задал почтовый ящик завершения, то в этот почтовый ящик посылается сообщение о том, что данный процесс удален. Для отсоединенных процессов, созданных операционной системой МОС ЭП, сообщение об окончании посылается системному диспетчеру заданий;

) удаляется область управления виртуальным адресным пространством процесса. (область управления содержит память, которую операционная система МОС ВП распределяет и использует для нужд данного процесса);

) удаляется вся построенная операционной системой мос вп информация о данном процессе.

8.7.1. Программа системного обслуживания ¤DELPRC

Процесс может удалить себя или другой процесс в какой-либо момент времени в зависимости от ограничений, которые изложены в п. 8.4.1. Удаляет процесс программа системного обслуживания ¤DELPRC.

Пример.

```
CYGNUS: .ASCID  /CYGNUS/        дескриптор для
                                имени процесса
```

```
¤DELPRC_S -
        PRCNAM=CYGNUS
```

Обычно нет необходимости явно вызывать программу системного обслуживания ¤DELPRC, так как подпроцесс автоматически удаляется, когда заканчивается выполнение образа (или когда поток команд для командного интерпретатора достигает конца файла).

В качестве альтернативы удалению процесса как средству остановки образа можно использовать программу системного обслуживания ¤FORCEX, чтобы создать принудительное завершение работы образа, работающего в процессе (п. 8.6.4.).

8.7.2. Почтовые ящики завершения

Почтовые ящики завершения позволяют процессу определить, когда и при каких условиях удаляется созданный им процесс. Программа системного обслуживания ¤CREPRC допускает в качестве аргумента номер устройства почтового ящика. Когда созданный процесс удаляется, этот почтовый ящик получает сообщение о его завершении.

Первое слово сообщения о завершении содержит символьную константу MSG¤_DELPROC, которая указывает, что данное сообщение является сообщением о завершении. Второе длинное слово сообщения о завершении содержит значение конечного состояния образа. Остаток сообщения содержит системную учетную информацию, используемую диспетчером заданий, посылаемую в системный учетный файл регистрации. Ч.LIN 0 если требуется, создающий процесс может определить номер идентификатора

фикации удаляемого процесса из блока состояния ввода-вывода. Второе длинное слово IOSB содержит номер идентификации удаляемого процесса.

Почтовый ящик завершения нельзя размещать в памяти, разделяемой многими процессорами.

9. Служебные программы таймера и преобразования времени

Часто требуются программные действия, запланированные по времени. В операционной системе МОС ВП образ может планировать некоторые события на задачное время или через заданный интервал времени. Существуют следующие служебные программы таймера и преобразования времени:

1) получить время (¤GETTIM);

2) преобразовать время из системного формата в числовой (¤NUMTIM);

3) преобразовать время из системного формата в строку символов в коде КОИ-8 (¤ASCTIM);

4) преобразовать строку символов в коде КОИ-8 во время в системном формате (¤BINTIM);

5) установить таймер (¤SETIMR);

6) отменить запрос к таймеру (¤CANTIM);

7) запланировать пробуждение (¤SCHDWK);

8) отменить пробуждение (¤CANWAK);

9) установить системное время (¤SETIME).

Служебные программы таймера можно использовать для того, чтобы планировать, преобразовывать или отменять собы-

тия. Например, их можно использовать для следующих целей:

1) запланировать установку флага события, или постановку в очередь асинхронного системного прерывания (AST) для текущего процесса, или отменить выставленный запрос, если он еще не был удовлетворен;

2) запланировать запрос на пробуждение процесса, находящегося в состоянии спячки, или отменить выставленный запрос на пробуждение, если он еще не был удовлетворен;

3) установить или изменить текущее системное время, если вызывающий процесс имеет соответствующие пользовательские привилегии.

Служебные программы таймера требуют задания времени в 64-битовом формате. Для работы с временами, представленными в других форматах, можно использовать служебные программы преобразования времени, которые позволяют:

1) получить текущую дату и время в виде строки символов в коде КОИ-8 или в системном формате;

2) преобразовывать строку символов в коде КОИ-8 в формат системного времени;

3) преобразовывать значение системного времени в строку символов в коде КОИ-8;

4) преобразовывать время из системного формата в числовые величины.

В данном разделе описан формат системного времени и служебные программы, которые используют его, а также даны примеры планирования программных действий при помощи служебных программ таймера.

9.1. Формат системного времени

Операционная система МОС ВП использует текущую дату и время в 64-битовом формате. Значение времени - это двоичное число, измеряемое в единицах, равных 100 нс (наносекунд), и представляющее смещение от системной базовой даты и времени, а именно 17 ноября 1858 г., 00 ч. 00 м. (смитсонианская базовая дата и время для астрономического календаря). Значения времени должны передаваться программам системного обслуживания или возвращаться ими как адрес квадрослова, содержащего время в 64-битовом формате. Значение времени может представлять собой одно из двух:

1) абсолютное время, которое представляет определенную дату и время дня. Абсолютное время всегда имеет положительное значение (или равно нулю);

2) дельта-время, которое является смещением некоторой будущей даты или времени относительно текущего момента времени. Дельта-время всегда выражается отрицательной величиной.

Если в качестве адреса значения времени задан нуль, операционная система МОС ВП будет обеспечивать текущую дату и время.

9.2. Получение текущей даты и времени

Текущее время в системном форчате можно получить при помощи программы системного обслуживания $GETTIM, которая помещает время в буфер.

Пример 1.

```
TIME:   .BLKQ   1                   буфер для времени


        ¤GETTIM_S -
                TIMARD=TIME         получить время
```

В примере 2 программа системного обслуживания ¤ASCTIM преобразует время, заданное в системном формате, в строку символов в коде КОИ-8 и возвращает эту строку в 23-байтовый буфер.

Пример 2.

```
ATIMENOW:                           дескриптор для времени
                                    в коде кои-8
        .LONG   23                  длина буфера
        .ADDRESS -                  адрес буфера
                TIMESTR
TIME_VALUE:                         64-битовое значение вре-
                                    мени, подлежащего преоб-
                                    разованию
        .BLKQ   1
TIMESTR:
        .BLKB   23                  возвращаемые 23 байта


        ¤ASCTIM_S -
```

```
TIMBUF=ATIMENOW,-
TIMADR=TIME_VALUE
```

Поскольку адрес 64-битового значения времени не был указан, то использовалось значение по умолчанию - нуль.

Строка, возвращаемая служебной программой имеет следующий формат:

```
DD-MMM-YYYY     HH:MM:SS.CC
```

Где DD - день месяца;

MMM - месяц (трехсимвольная буквенная аббревиатура);

YYYY - год;

HH:MM:SS.CC - время в часах, минутах, секундах и сотых секунды.

9.3. Получение абсолютного времени в системном формате

Программа системного обслуживания ¤BINTIM является противоположной по назначению программе системного обслуживания ¤ASCTIM. Служебной программе передается время в формате строки в коде КОИ-8 и она преобразует эту строку в значение времени в 64-битовом формате. Данное возвращаемое значение можно затем использовать как вход для служебной программы планирования по таймеру.

При задании буфера строки символов в коде кои-8 можно опустить некоторые поля, и служебная программа будет использовать для этих полей текущую дату или время. Так, если требуется сделать запрос к таймеру, независящий от даты, следует форматировать буфер для служебной программы ¤BINTIM так, как это показано в примере. Необходимо включить два дефиса, которые обычно входят в поле даты, и по крайней мере один пробел перед полем времени.

Пример.

```
ASCII_NOON:
        .ASCID  /-- 12:00:00.00/        дескриптор для времени 12 часов дня в коде КОИ-8
BINARY_NOON:                            буфер для двоичного времени 12 часов дня
        .BLKQ   1

        ¤BINTIM_S -                     преобразовать время
                TIMBUF=ASCII_NOON
                TIMADR=BINARY_NOON
```

9.4. Получение дельта-времени в системном формате

Программа системного обслуживания ¤BINTIM преобразует также строки символов в коде КОИ-8 в значения

дельта-времени, используемые как входные значения для служебных программ таймера. Буфер для строк в коде КОИ-8, представляющих дельта-время, имеет вид:

DDDD HH:MM:SS.CC

Если задается дельта-время для текущего дня, то первое поле, представляющее количество дней, должно быть равно нулю.

Следующий пример показывает, как использовать служебную программу @BNITIM для получения дельта-времени в системном формате:

Пример.

```
ATENMIN:
        .ASCID  /0 00:10:00.00/     дескриптор для времени
                                    "10 минут" в коде кои-8
BTENMIN:
        .BLKQ   1                   буфер для двоичного вре-
                                    мени "10 минут"


        @BINTIM_S                   преобразовать время
                TIMBUF=ATENMIN,-
                TIMADR=BTENMIN
```

Пользователь языка макроассемблер может также задавать приближенные значения дельта-времени, используя директивы .LONG для того, чтобы представить значения времени в единицах, равных 100 наносекундам. Эта арифметика основана на

следующей формуле:

1 секунда = 10 миллионов * 100 наносекунд (1)

Например, следующая директива определяет значение дельта-времени, равное 5 секундам.

Пример.

```
FIVESEC:        .LONG   -10*1000*1000*5, -1   5 секунд
```

Для наглядности значение 10 миллионов записано как 10*1000*1000. Значение дельта-времени отрицательно.

Однако, при использовании такой нотации значение времени ограничено максимальным числом в единицах, равных 100 наносекундам, помещающимся в длинное слово. В единицах времени это число немногим больше 7 минут.

9.5. Запросы к таймеру

Запросы к таймеру, сделанные программой системного обслуживания ¤SETIMR, ставятся в очередь, откуда они выбираются для обработки в том порядке, в каком истекают заданные в них времена. Количеством запросов, которые процесс может поставить в очередь к таймеру, управляет квота TQELM.

При вызове программы системного обслуживания ¤SETIMR можно задавать любое абсолютное время, либо значение дельта-времени. В зависимости от того, как требуется обработать запрос, можно указывать один из двух или оба следующих аргумента:

1) номер флага события, который требуется установить, когда истечет заданное время. (Если флаг события не указан, операционная система МОС ВП устанавливает флаг события 0);

2) адрес подпрограммы обслуживания асинхронных системных прерываний (AST), которая должна быть выполнена по истечении заданного времени.

По желанию можно указать идентификатор для запроса к таймеру. Этот идентификатор можно использовать для того, чтобы, если требуется, отменить данный запрос. Кроме того, идентификатор запроса передается как AST-параметр подпрограмме обслуживания асинхронных системных прерываний (AST), если она задана, так что подпрограмма обслуживания AST может распознать данный запрос к таймеру.

В примерах 1 и 2 показаны запросы к таймеру, использующие флаги событий и асинхронные системные прерывания (разделы 4,5).

Пример 1.

Установка флага события.

```
A30SEC: .ASCID  /0 00:00:30.00/   дескриптор для дельта-
                                  времени "30 секунд" в
                                  коде КОИ-8
B30SEC: .BLKQ   1                 квадрослово для хранения
                                  преобразованного (двоич-
                                  ного) дельта-времени

        $BINTIM_S                 преобразовать в двоичное
                TIMBUF=A30SEC,-
                TIMADR=B30SEC
```

```
        BSBW    ERROR
        $SETIMR_S -                 установить время ожидания
                EFN=#4, -
                DAYTIM=B30SEC
        BSBW    ERROR               вызвать подпрограмму об-
                                    работки ошибок
        $WAITFR_S -                 ждать 30 секунд
        EFN=#4
        BSBW    ERROR
```

Примечания:

1. Вызов программы $SETIMR требует, чтобы флаг события 4 был установлен через 30 секунд (выраженных в двоичном виде в квадрослове B30SEC).

2. Программа системного обслуживания $WAITFR переводит процесс в состояние ожидания до тех пор, пока не будет установлен данный флаг события.

Пример 2.

Использование подпрограммы обслуживания AST.

```
ANOON:  .ASCID  /-- 12:00:00.00/   дескриптор для времени
                                   "12 часов дня" в коде
                                   КОИ-8
BNOON:  .BLKQ   1                  содержит преобразован-
                                   ное (двоичное) время
                                   "полдень"
```

```
        $BINTIM_S -                 преобразовать в двоичное
                TIMBUF=ANOON, -
                TIMADR=BNOON
        BSBW    ERROR
$SETIMR_S -
                DAYTIM=BNOON, -     установить таймер на пол-
                                    день
                ASTADR=ASTSERV,-;   указать подпрограмму AST
                REQIDT=#12      ;   идентификатор запроса на
                                    полдень как AST-параметр
        BSBW    ERROR


        RET
$ENTRY  ASTSERV,^M<>                маска входа подпрограммы
                                    AST
 CMPL   #12,4(AP)                   это запрос "полдень"?
 BNEQ   10$                         если нет, обработать дру-
                                    гой тип

                                    обработать запрос AST
                                    "полдень"
 RET
```

10¤: обработать другие типы
запросов

RET

Примечания:

1. Вызов программы ¤BINTIM преобразует строку в коде КОИ-8, представляющую 12.00 дня, в системный формат. Значение, возвращаемое в BNOON, используется как входной параметр программы системного обслуживания ¤SETIMR.

2. Подпрограмма AST, указанная в запросе ¤SETIMR, будет выполняться по истечении заданного времени, то есть в 12.00 дня. Аргумент REQIDT идентифицирует данный запрос к таймеру. (этот аргумент передается как aST-параметр и хранится в списке аргументов со смещением 4. (см. раздел 5). Процесс продолжает выполнение; когда заданное время наступит, он прервется доставкой AST. Если текущее время больше, чем полдень, то таймер сработает сразу же.

3. Данная подпрограмма обслуживания AST проверяет параметр, переданный аргументом REQIDT, чтобы узнать (в данном примере), будет ли она обслуживать запрос к таймеру "12.00 дня" или другой тип запроса (определяемый другим значением идентификатора запроса). После выполнения подпрограммы обслуживания AST процесс продолжает выполнение с той точки, в которой он был прерван.

9.5.1. Отмена запросов к таймеру

Программа системного обслуживания ¤CANTIM отменяет те запросы к таймеру, которые еще не обработаны. Данные запросы удаляются из очереди к таймеру. Отмена базируется на идентификации запроса, указанной в запросе к таймеру. Например, чтобы отменить запрос, показанный в примере 2 (подраздел 9.5.), необходимо сделать следующий вызов программы ¤CANTIM.

Пример.

```
¤CANTIM_S       REQIDT=#12
```

Если такой же идентификатор был присвоен нескольким запросам к таймеру, то будут отменены все запросы с этим идентификатором. Если пользователь не указал аргумент REQIDT, то все запросы этого пользователя будут отменены.

9.6. Запланированные пробуждения от спячки

В примере 1 (см. подраздел 9.5) показан подпроцесс, который переводит себя в состояние ожидания при помощи служебных программ ¤SETIMR и ¤WAITFR; другой способ сделать процесс неактивным заключается в переводе его в состояние спячки. Процесс переводит себя в состояние спячки при помощи программы системного обслуживания ¤HIBER; возвращение из спячки осуществляется при помощи запроса на пробуждение, который срабатывает немедленно, если используется программа системного обслуживания ¤WAKE, или в зависимости от времени, если используется программа системного обслуживания

¤SCHDWK (подраздел 8.5).

В примере 1 показан процесс, который планирует для себя пробуждение прежде, чем перейти в спячку.

Пример 1.

```
ATENSEC:                        дескриптор для времени
        .ASCID  /0 00:00:10.00/ ожидания 10 секунд
BTENSEC:
        .BLKQ   1               сохранить двоичное значе-
                                ние десяти секунд

        ¤BINTIM_S               преобразовать время
                TIMBUF=ATENSEC, -
                TIMADR=BTENSEC
        ¤SCHDWK_S -             запланировать пробуждение
                DAYTIM=BTENSEC
        ¤HIBER_S                уснуть на 10 секунд
```

Спячка и пробуждение подробно описаны в разделе 8. Процесс, имеющий соответствующую привилегию, может запрашивать пробуждение или планировать пробуждение для другого процесса; таким образом, взаимосвязанные процессы могут синхронизировать свои действия, используя перевод в состояние спячки и запланированные пробуждения. Более того, при использовании в программе программы системного обслуживания ¤SCHDWK можно указать, что запрос на пробуждение должен повторяться через фиксированные интервалы времени.

9.6.1. Отмена запланированных пробуждений

Запросы на запланированное пробуждение, если они выставлены, но еще не обработаны, можно отменить при помощи программы системного обслуживания ¤CANWAK.

В примере 1 показано планирование запросов на пробуждение процесса CYGNUS и последующая отмена этих пробуждений. В данном примере программа системного обслуживания ¤SCHDWK задает дельта-время 1 минуту, и интервал времени 1 минуту; пробуждение повторяется каждую минуту, пока эти запросы не будут отменены.

Пример 1.

```
CYGNUS: .ASCID  /CYGNUS/          дескриптор для имени
                                  процесса
ONE_MIN:
        .ASCID  /0 00:01:00.00/   дескриптор для 1 минуты
                                  (дельты)
INTERVAL:
        .BLKQ   1                 8 байтов для хранения
                                  двоичной 1 минуты


        ¤BINTIM_S -               преобразовать в двоичное
                TIMBUF=ONe_MIN, -
                TIMADR=INTERVAL


        ¤SCHDWK_S -               пробуждать каждую минуту
```

```
00152-01 97 06

        PRCNAM=CYGNUS, -
        DAYTIM=INTERVAL, -
        REPTIM=INTERVAL


¤CANWAK_S -                    отменить пробуждение
        PRCNAM=CYGNUS
```

9.7. Числовое время и время в коде КОИ-8

Программа системного обслуживания ¤NUMTIM преобразует время, выраженное в системном формате, в двоичные целые значения. Эта программа возвращает каждую компоненту времени (год, месяц, день, час и т.д.) в отдельное слово буфера длиной 7 слов.

Чтобы отформатировать время в строку в коде КОИ-8 для включения его в выходной поток, можно использовать программу системного обслуживания ¤ASCTIM. Служебная программа ¤ASCTIM принимает в качестве аргумента адрес квадрослова, содержащего время в системном формате, и возвращает дату и время в формате КОИ-8.

Если желательно включить дату и время в символьные строки, которая содержит дополнительные данные, можно форматировать выходную строку при помощи программы ¤FAO. Программа системного обслуживания ¤FAO преобразует двоичные величины в коды КОИ-8 и подставляет результаты в символьные

строки в соответствии с директивами, содержащимися во входной управляющей строке. Среди них есть директивы !%T и !%D, которые преобразуют значение времени, содержащееся в квадрослове, в строку в коде КОИ-8 и подставляет результат в выходную строку.

9.8. Установка системного времени

Программа системного обслуживания ¤SETIME дает возможность пользователю, имеющему привилегии OPER и LOG_IO, устанавливать текущее системное время. Можно задавать новое системное время (используя аргумент TIMADR), либо заново разметить текущее системное время при помощи аппаратных процессорных часов (опуская аргумент TIMADR). Если задается время, то оно должно быть значением абсолютного времени; значения дельта-времени (отрицательные) не допускаются.

Системное время устанавливается при каждой первоначальной загрузке операционной системы МОС ВП. Обычно нет необходимости изменять системное время между перезагрузками системы, однако, при определенных условиях может потребоваться изменить системное время без перезагрузки. Например, можно задавать новое системное время для того, чтобы синхронизировать два процессора, или чтобы отрегулировать расхождение между стандартным и световым временами(при переходе на летнее время).

Программа системного обслуживания ¤SETIME вызывается командой диалогового командного языка DCL SET TIME.

Если процесс выдал запрос на дельта-время, и системное время изменилось, то интервал, остающийся до удовлетворения запроса, не изменяется; то есть, запрос выполняется после того, как истечет заданное время. Если процесс выдал запрос на абсолютное время, и системное время изменилось, то запрос выполняется в указанное время относительно нового системного времени.

10. Служебные программы обработки кода состояния

Обработчик кода состояния - это программа, которой передается управление при возникновении исключительных ситуаций. Исключительной ситуацией является событие, которое обнаруживается аппаратным или программным обеспечением и которое прерывает выполнение образа. Примером исключительной ситуации может быть арифметическое переполнение и сбои на недействительных кодах операции или операндах.

В том случае, если программа, имея информацию о конкретной исключительной ситуации, способна предпринять какие-либо корректирующие действия, можно написать и указать подпрограмму обработки кода состояния. Эта подпрограмма обработки кода состояния, которая будет получать управление при возникновении любой исключительной ситуации, может осуществить проверку на заданные исключительные ситуации.

Если возникла какая-то исключительная ситуация, и пользователь не указал подпрограмму обработки кода состояния, то управление передается подпрограмме обработки кода состояния, принятой по умолчанию операционной системой мос вл. Если исключительная ситуация является результатом неисправимой ошибки, то принятая по умолчанию подпрограмма обработки кода состояния выдает диагностическое сообщение и организует завершение образа, который явился причиной данной исключительной ситуации.

В данном разделе описано, как работает механизм обра-

ботки кода состояния в МОС ВП, и даны пояснения к написанию подпрограммы обработки кода состояния. При написании подпрограммы обработки кода состояния используются следующие системные программы:

1) установить вектор исключительной ситуации (¤SETEXV);

2) установить режим исключительной ситуации по ошибке программ системного обслуживания (¤SETSFM);

3) развернуть стек вызовов для поиска подпрограммы обработки кода состояния (¤UNWIND);

4) об'явить подпрограмму обработки изменения режима или режима совместимости (¤DCLCMH).

10.1. Типы исключительных ситуаций

Исключительные ситуации могут создаваться каким-либо из следующих средств:

1) аппаратное обеспечение;

2) программное обеспечение;

3) ошибки программ системного обслуживания.

Исключительные ситуации, создаваемые аппаратным обеспечением, всегда приводят к условиям, требующим специальных действий, если требуется продолжить выполнение программы.

Исключительные ситуации, создаваемые программным обеспечением, могут привести к условиям ошибки или предупреждения (документы [14] и [15]).

Исключительные ситуации по ошибке программ системного обслуживания возникают в том случае, если из вызова прог-

раммы системного обслуживания возвращается код состояния простой или серьезной ошибки. Обработка возврата ошибки из программ системного обслуживания при помощи механизма обработки кода состояния предпочтительнее, чем другие методы проверки ошибки. Чтобы иметь возможность обрабатывать исключительные ситуации, созданные ошибками программ ситемного обслуживания, необходимо включить режим исключительной ситуации по ошибке программ системного обслуживания при помощи программы ¤SETSFM.

Пример.

```
¤SETSFM_S        ENBFLG = #1
```

Режим исключительной ситуации по ошибке программ системного обслуживания первоначально выключен и может быть включен или выключен в любой момент выполнения образа.

В табл. 36 включены общие коды состояния, вызванные исключительными ситуациями. В первом столбце перечислены имена кодов состояний. Второй столбец более полно описывает состояние, давая информацию о типе, значении и аргументах, относящихся к каждому состоянию (см. документы [5] и [6]). Значение состояния исключительной ситуации - это краткое описание каждого состояния. Перечислены аргументы для подпрограммы обработки кода состояния (если таковые имеются), которые обеспечивают специальную информацию о состоянии.

Таблица 36

Коды состояния исключительных ситуаций

Имя кода состояния	Пояснение
SSB_ACCVIO	Тип: сбой описание: нарушение доступа Аргументы: 1. Причина нарушения доступа. это маска следующего формата: бит 0 - тип нарушения доступа; 0 - код защиты элемента таблицы страниц не допускает предполагаемого доступа; 1 - нарушение длины P0LR, P1LR или SLR (длины области программы, области управления или системы); бит 1 - ссылка на элемент таблицы страниц; 0 - указанный виртуальный адрес не досту-

Продолжение табл. 36

Имя кода состояния		Пояснение
		пен; 1 - соответствующий элемент таблицы страниц не доступен; бит 2 - предполагаемый доступ; 0 - чтение; 1 - модификация.
		2. Виртуальный адрес, к которому была попытка получить доступ
SS¤_ARTRES	Тип:	ловушка
	Описание:	прерывание по недействительной арифметике
	Аргументы:	нет
SS¤_ASTFLT	Тип:	ловушка
	Описание:	неверный стек при попытке доставить AST
	Аргументы:	1. Значение указателя стека при сбое.
		2. AST-параметр давшего ошибку AST.

Продолжение табл. 36

Имя кода состояния		Пояснение
		3. Программный счетчик (PC) в момент прерывания доставки AST.
		4. Длинное слово состояния процессора (PSL) в момент прерывания доставки AST.
		5. PC, к которому должно быть доставлено AST.
		6. PSL, к которому должно быть доставлено AST
SS¤_BREAK	Тип:	сбой
	Описание:	встретилась инструкция останов
	Аргументы:	нет
SS¤_CMODSUPR	Тип:	ловушка
	Описание:	встретилась инструкция изменить режим на режим супервизора
	Аргументы:	код режима изменения. Возможные значения от минус 32768 до 32767
SS¤_CMODUSER	Тип:	ловушка
	Описание:	встретилась инструкция изме-

Продолжение табл. 36

Имя кода состояния		Пояснение
		нить режим на режим пользова-
		теля
	Аргументы:	код режима изменения. Возмож-
		ные значения от минус 32768
		до 32767
SS¤_COMPAT	Тип:	сбой
	Описание:	исключительная ситуация режима
		совместимости. Данное условие
		исключительной ситуации может
		возникнуть только при работе
		в режиме совместимости
	Аргументы:	тип исключительной ситуации
		режима совместимости. Возмож-
		ные значения:
		0 - исключительная ситуация
		недействительной инструкции
		1 - выполнялась инструкция
		BPT;
		2 - выполнялась инструкция
		IOT;
		3 - выполнялась инструкция
		EMT;

Продолжение табл. 36

Имя кода состояния	Пояснение
	4 - выполнялась инструкция TRAP;
	5 - неверная инструкция;
	6 - сбой по нечетному адресу;
	7 - прерывание TBIT.
SS¤_DECOVF	Тип: ловушка
	Описание: десятичное переполнение
	Аргументы: нет
SS¤_FLTDIV	Тип: ловушка
	Описание: деление числа с плавающей запятой или десятичного числа на нуль
	Аргументы: нет
SS¤_FLTDIV_F	Тип: сбой
	Описание: сбой при делении числа с плавающей запятой на нуль
	Аргументы: нет
SS¤_FLTOVF	Тип: ловушка
	Описание: переполнение числа с плавающей запятой
	Аргументы: нет

Продолжение табл. 36

Имя кода состояния	Пояснение	
SS¤_FLTOVT_F	Тип:	сбой
	Описание:	сбой при переполнении числа с плавающей запятой
	Аргументы:	нет
SS¤_FLTUND	Тип:	ловушка
	Описание:	переполнение по нижней границе диапазона числа с плавающей запятой
	Аргументы:	нет
SS¤_FLTUND_F	Тип:	сбой
	Описание:	сбой при переполнении по нижней границе диапазона числа с плавающей запятой
	Аргументы:	нет
SS¤_INTDIV	Тип:	ловушка
	Описание:	деление целого числа на нуль
	Аргументы:	нет
SS¤_INTOVF	Тип:	ловушка
	Описание:	переполнение целого числа
	Аргументы:	нет
SS¤_OPCCUS	Тип:	сбой
	Описание:	сбой на коде операции, не-

Продолжение табл. 36

Имя кода состояния		Пояснение
		допустимом для пользователя
	Аргументы:	нет
SS$_OPCDEC	Тип:	сбой
	Описание:	сбой на недопустимом коде операции
	Аргументы:	нет
SS$_PAGRDERR	Тип:	сбой
	Описание:	ошибка при чтении, возникшая при попытке считать с диска ошибочную страницу
	Аргументы:	причина, по которой невозможна пересылка. Эта маска следующего формата: бит 0 - 0; бит 1 - ссылка на элемент таблицы страниц; 0 - указанный виртуальный адрес недопустим; 1 - соответствующий элемент таблицы страниц не допустим;

Продолжение табл. 36

Имя кода состояния		Пояснение
		бит 2 - предполагаемый доступ; 0 - чтение; 1 - модификация
SS$_RADRMOD	Тип:	сбой
	Описание:	попытка использовать недопустимый режим адресации
	Аргументы:	нет
SS$_ROPRAND	Тип:	сбой
	Описание:	попытка использовать недопустимый операнд
	Аргументы:	нет
SS$_SSFAIL	Тип:	сбой
	Описание:	ошибка программы системного обслуживания (если разрешен режим исключительной ситуации по ошибке программы системного обслуживания)
	Аргументы:	код состояния, возвращаемый из программы системного обслуживания (R0)
SS$_SUBRNG	Тип:	ловушка

Продолжение табл. 36

Имя кода состояния		Пояснение
	Описание:	ловушка по диапазону индексов
	Аргументы:	нет
SS¤_TBIT	Тип:	сбой
	Описание:	бит трассировки не разрешен следующей инструкции
	Аргументы:	нет

PC и PSL, которые обычно включены в сигнальный массив, в данный список аргументов не включаются. Указатель стека для режима доступа, получившего данную исключительную ситуацию, устанавливается в его начальное значение.

Если в пользовательском или супервизорном режиме при помощи программы ¤DCLCMH была об'явлена подпрограмма обработки режима изменения, то она получает управление при возникновении соответствующего прерывания.

Если при помощи программы ¤DCLCMH была об'явлена подпрограмма обработки режима совместимости, то она получает управление при возникновении данного сбоя.

10.1.1. Программа обработки изменения режима и режима совместимости

При помощи обычного механизма обработки кода состояния, описываемого в данном разделе, можно специальным образом обрабатывать два типа аппаратных исключительных ситуаций:

1) прерывания, вызванные инструкциями изменить режим на режим пользователя или изменить режим на режим супервизора;

2) сбои режима совместимости.

Для того, чтобы задать подпрограммы, которые будут получать управление при возникновении одной из этих ситуаций, можно использовать программу системного обслуживания ¤DCLCMH.

10.2. Задание обработчиков кода состояния

Подпрограммы обработки кода состояния, которые должны получать управление в случае возникновения исключительной ситуации, можно указывать двумя способами:

1) задать адрес маски входа обработчика кода состояния в первом длинном слове кадра вызова;

2) об'явить обработчик исключительных ситуаций при помощи программы системного обслуживания ¤SETEXV.

Первый из этих методов является более предпочтительным при задании обработчика кода состояния для отдельного образа. Векторные обработчики следует использовать для спе-

циальных целей, например, для написания отладчиков. Программист, пишущий на языке макроассемблер, может при помощи единственной инструкции "переслать адрес" поместить адрес обработчика кода состояния в длинное слово, определяемое указателем текущего кадра (FP).

Пример.

```
MOVAB   HANDLER,(FP)
```

Программист, использующий язык высокого уровня, может вызвать из исполнительной библиотеки программу LIB$ESTABLISH (см. документы [9] - [13]). Однако, некоторые языки обеспечивают доступ к обработке кода состояния как часть языка.

Каждая процедура в стеке вызовов может об'явить обработчик кода состояния.

Программа системного обслуживания $SETEXV позволяет задать адреса для первичного обработчика исключительной ситуации, для вторичного обработчика исключительной ситуации и для окончательного обработчика исключительной ситуации. Обработчики можно указывать для каждого режима доступа. Первичный вектор исключительной ситуации резервируется для отладчика. При использовании векторного обработчика нужно предусматривать в нем обработку всех исключительных ситуаций, которые могут возникнуть в данном режиме доступа.

Адрес, равный 0, в первом длинном слове кадра вызова процедуры или в векторе исключительной ситуации означает, что для данного кадра вызова или вектора нет обработчика кода состояния.

10.3. Диспетчер исключительных ситуаций

При возникновении исключительной ситуации управление передается подпрограмме диспетчеризации исключительных ситуаций, входящей в операционную систему МОС ВП. Диспетчер исключительных ситуаций организует поиск подпрограммы обработки исключительных ситуаций в следующем порядке:

1) первичный вектор исключительной ситуации для того режима доступа, в котором работала данная программа при возникновении исключительной ситуации;

2) вторичный вектор исключительной ситуации для того режима доступа, в котором работала данная программа при возникновении исключительной ситуации;

3) адрес обработчика условия, заданный в стеке вызова процедуры для того режима доступа, в котором работала данная программа при возникновении исключительной ситуации. Кадры вызова в стеке сканируются в обратном порядке, при этом для обращения к предыдущему кадру вызова используется сохраняемый в каждом кадре вызова указатель кадра;

4) окончательный вектор исключительной ситуации для того режима доступа, в котором работала данная программа при возникновении исключительной ситуации.

Поиск заканчивается, когда диспетчер находит обработчик кода состояния. Если диспетчер не может найти заданный пользователем обработчик кода состояния, то он вызывает обработчик кода состояния, адрес которого хранится в окончательном векторе исключительной ситуации. Если образ был инициирован командным интерпретатором, то окончательный

вектор указывает на всеобъемлющий обработчик кодов состояния. Всеобъемлющий обработчик кодов состояния выдает сообщение и либо продолжает выполнение программы, либо организует выход образа в зависимости от того, является ли данное состояние соответственно предупреждающим или ошибочным состоянием.

Всеобъемлющий обработчик можно вызвать двумя способами:

1) если окончательный вектор исключительной ситуации возвращает управление диспетчеру, или окончательный вектор исключительной ситуации пуст, диспетчер вызывает всеобъемлющий обработчик кодов состояния и заканчивается с кодом возврата SS$_NOHANDLER;

2) если диспетчер исключительных ситуаций обнаруживает нарушение доступа, он вызывает всеобъемлющий обработчик кодов состояния и заканчивается с кодом возврата SS$_ACCVIO.

На рис. 41 показана организация диспетчером исключительных ситуаций поиска обработчика кода состояния в стеке вызовов.

Поиск обработчику кода состояния

в стеке вызовов

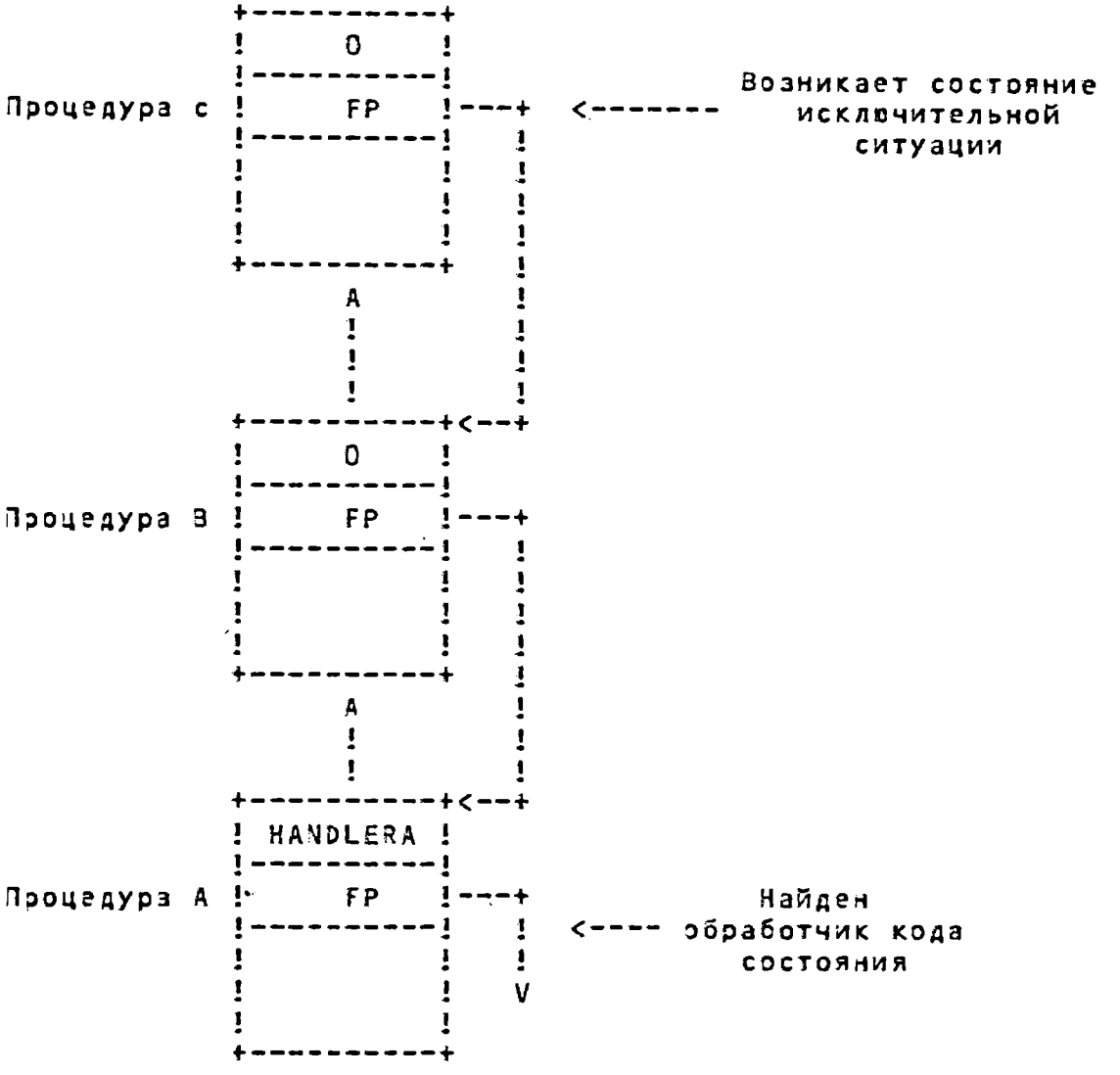


Рис. 41

Примечания:

1. Изображение стека вызовов показывает последовательность вызовов: процедура А вызывает процедуру В, процедура В вызывает процедуру С. В процедуре А организован обработчик кода состояния.

2. Исключительная ситуация возникает при выполнении процедуры С. Диспетчер исключительных ситуаций организует

поиск обработчика кода состояния.

3. После проверки наличия обработчика кода состояния, об'явленного в векторах исключительной ситуации, диспетчер (в предположении, что в векторах ничего не указано для данного процесса) смотрит первое длинное слово в кадре вызова процедуры C. Значение 0 указывает на то, что обработчик кода состояния не задан. Диспетчер, используя указатель кадра (FP) в кадре вызова процедуры C, находит кадр вызова процедуры B. Здесь он снова не находит обработчик кода состояния и переходит к кадру вызова процедуры A.

4. Диспетчер находит обработчик кода состояния HANDLERA и передает ему управление.

10.4. Список аргументов, передаваемых обработчику кода состояния

Когда диспетчер находит обработчик кода состояния, он передает ему управление при помощи инструкции CALLG. Список аргументов, передаваемых обработчику кода состояния, имеет структуру стека и содержит адреса двух массивов аргументов, как показано на рис. 42 (пп. 10.4.1 и 10.4.2).

При помощи макрокоманды $SHFDEF можно определить для обращения к данным аргументам следующие символические имена, приведенные в табл.37.

Таблица 37

Символическое имя	Значение
CHF$L_SIGARGLST	Адрес сигнального массива
CHF$L_MCHARGLST	Адрес массива механизма
CHF$L_SIG_ARGS	Количество сигнальных аргументов
CHF$L_SIG_NAME	Имя кода состояния
CHF$L_SIG_ARG1	Первый заданный сигнальный аргумент
CHF$L_MCH_ARGS	Количество аргументов механизма
CHF$L_MCH_FRAME	Адрес кадра вызова, содержащего обработчик кода состояния
CHF$L_MCH_DEPTH	Глубина в стеке кадра вызова, содержащего обрабочик кода состояния
CHF$L_MCH_SAVR0	Сохраняемый регистр R0
CHF$L_MCH_SAVR1	Сохраняемый регистр R1

10.4.1. Аргументы сигнального массива

Сигнальный массив содержит следующие величины, описывающие данное состояние:

1) имя кода состояния - символьное значение, присвоенное данному коду состояния. Возможные коды состояния исключительных ситуаций и их символические определения приведены в табл. 36;

2) аргументы - специальная информация, относящаяся к данному состоянию. Аргументы, обеспечиваемые каждым состоянием, также приведены в табл. 36;

3) PC - программный счетчик на момент возникновения исключительной ситуации (сбой или ловушка) он может содержать либо адрес инструкции (для сбоя), вызвавшей исключительную ситуацию, либо адрес следующей инструкции (для ловушки);

4) PSL - длинное слово состояния процессора на момент возникновения исключительной ситуации.

10.4.2. Аргументы массива механизма

Массив механизма описывает контекст, в котором возникла исключительная ситуация. Он обеспечивает следующие аргументы:

1) кадр вызова, содержащий обработчик кода состояния - содержимое регистра (FP) указателя кадра вызова процедуры, которая организовала данный обработчик кода состояния. Это адрес длинного слова, содержащего адрес обработчика кода состояния. Например, для стека вызовов, показанного на рис. 41, данный аргумент указывает на кадр вызова процедуры a;

2) глубина в стеке - номер кадра процедуры, организовавшей данный обработчик кода состояния, относительно кадра процедуры, в которой возникла исключительная ситуация. Глубина определяется следующим образом:

-3 - обработчик кода состояния был об'явлен в окончательном векторе исключительной ситуации;

-2 - обработчик кода состояния был об'явлен в первичном векторе исключительной ситуации;

-1. - обработчик кода состояния был об'явлен во вторичном векторе исключительной ситуации;

0 - обработчик кода состояния был об'явлен в кадре, который был активным в момент возникновения исключительной ситуации;

1 - обработчик кода состояния был об'явлен вызывателем кадра, активного в момент возникновения исключительной ситуации;

2 - обработчик кода состояния был об'явлен вызывателем вызывателя того кадра, который был активным в момент возникновения исключительной ситуации;

и т.д.

Например, для стека вызовов, показанного на рис. 41, аргумент "глубина" передаваемый обработчику HANDLERA, будет иметь значение 2.

Обработчик кода состояния может определять по этому аргументу, необходимо ли ему обрабатывать данное состояние. Например, обработчик кода состояния может ничего не предпринимать, если исключительная ситуация, создавшая это состояние, возникла в кадре, не организовавшем данный обработчик;

3) R0 - содержимое регистра R0 в момент возникновения исключительной ситуации;

4) R1 - содержимое регистра R1 в момент возникновения исключительной ситуации.

Список аргументов,

передаваемых обработчика кода состояния

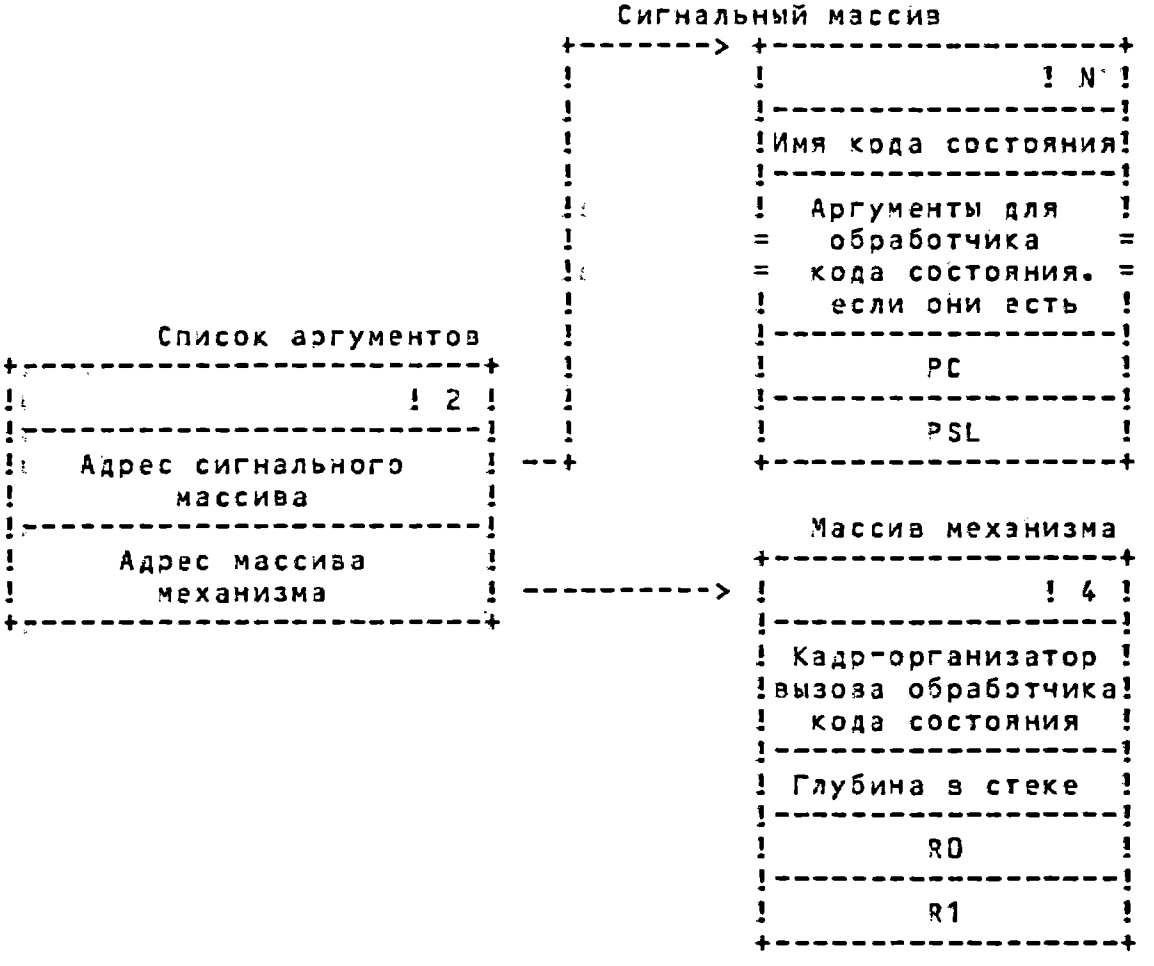


Рис. 42

10.5. Возможные действия обработчика кода состояния

После того, как подпрограмма обработки кода состояния определит природу исключительной ситуации, она может предпринять какие-либо действия в одном из трех направлений:

1) продолжить.

Независимо от того, в состоянии ли обработчик кода состояния решать проблему или нет, программа может пытаться продолжить выполнение. Обработчик помещает в R0 возвращаемый код состояния SS$_CONTINUE и при помощи инструкции RET возвращает управление диспетчеру. Если исключительной ситуацией был сбой, то снова выполняется инструкция, вызвавшая его, если исключительной ситуацией была ловушка, то управление передается инструкции, следующей за той, которая дала ловушку;

2) повторно сигнализировать.

Обработчик не может решить данную проблему, либо данное условие не относится к тем, которые он обрабатывает. В этом случае обработчик помещает в R0 возвращаемый код состояния SS$_RESIGNAL и при помощи инструкции RET возвращает управление диспетчеру исключительной ситуации. Если он находит другой обработчик кода состояния, то передает ему управление;

3) развернуть стек вызовов.

Обработчик кода состояния не может решить данную проблему, и выполнение нельзя продолжить. В этом случае обработчик для того, чтобы развернуть стек вызовов, использует программу системного обслуживания $UNWIND. После этого можно удалить из стека кадры вызовов и модифицировать порядок выполнения в соответствии с аргументами программы $UNWIND.

Первые два пути, которые может выбрать обработчик кода состояния приведены в следующем примере.

Пример.

```
        .ENTRY  PGMA,^M<>           маска входа процедуры а
MOVAB   HANDLERA,(FP)               об'явить обработчик ко-
                                    да состояния
        $SETSFM_S -
                ENBFLG=#1           разрешить исключительные
                                    ситуации по ошибке прог-
                                    рамм системного обслужи-
                                    вания
CALLG   ARGLIST,PGMB                вызвать процедуру в

.ENTRY  HANDLERA,^M<R2,R3,R4>       маска входа обработчика
                                    HANDLERA
        MOVL    CHF$L_SIGARGLST(AP),R4   получить адрес
                                    сигнальных аргументов
        CMPL    #SS$_SSFAIL, CHF$L_SIG_NAME(R4)
                                    это ошибка программы
                                    системного обслужива-
                                    ния?
        BNEQ    10$                 нет - повторно сигнали-
                                    зировать
                                    обработать исключитель-
                                    ную ситуацию по ошибке
                                    программы системного
                                    обслуживания
```

```
        MOVZWL  #SS¤_CONTINUE,R0    продолжить
        RET                         возврат в диспетчер ис-
                                    ключительных ситуаций
10¤:    MOVZWL  #SS¤_RESIGNAL,R0 ; повторно сигнализировать
        RET                         возврат в диспетчер
        .ENTRY  PGMB,^M<R2,R3,R4>   маска входа процедуры в
 MOVAB  HANDLERB,(FP)               об'явить обработчик ко-
                                    да состояния
                                +-- возникает ошибка прог-
              <--  --  -- --!       раммы системного обслу-
                                +-- живания
 .ENTRY  HANDLER3,^M<R2,R3,R4>      маска входа обработчика
                                    HANDLERB
        MOVL    CHF¤L_SIGARGLST(AP),R4 ; получить адрес
                                    сигнальных аргументов
        CMPL    #SS¤_BREAK,CHF¤L_SIG_NAME(R4); сбой по
                                    инструции останова?
        BNEQ    10¤                 нет, повторно сигнали-
                                    зировать
                                    да, обработать исключи-
                                    тельную ситуацию
        MOVZWL  #SS¤_CONTINUE,R0 ; продолжать
        RET                         возврат в диспетчер ис-
                                    ключительной ситуации
10¤:.   MOVZWL  #SS¤_RESIGNAL,R0 ;повторно сигнализировать
        RET                         возврат в диспетчер
```

Примечания:

1. Процедура A выполняется и организует обработчик кода состояния HANDLERA. HANDLERA предназначен для обработки исключительных ситуаций, возникающих при ошибках в вызовах программ системного обслуживания.

2. Во время своего выполнения процедура A вызывает процедуру B.

3. Процедура B организует обработчик кода состояния HANDLERB. HANDLERB предназначен для обработки исключительных ситуаций, вызванных сбоями в инструкциях останова.

4. Во время выполнения процедуры B возникает исключительная ситуация, вызванная ошибкой программы системного обслуживания.

5. Диспетчер исключительных ситуаций просматривает векторы исключительных ситуаций в поиске обработчика кода состояния (предполагается, что они ничего не определяют), а затем просматривает стек вызовов. Вызывается обработчик HANDLERB, которому передается код состояния SS¤_SSFAIL.

Поскольку HANDLERB обрабатывает только сбои в инструкциях останова, он помещает в R0 возвращаемое значение SS¤_RESIGNAL и возвращает управление диспетчеру исключительных ситуаций.

7. Диспетчер исключительных ситуаций возобновляет поиск обработчика кода состония.

8. HANDLERA обрабатывает исключительную ситуацию по ошибке программ системного обслуживания, исправляет ситуацию, помещает в R0 возвращаемое значение SS¤_CONTINUE и возвращает управление диспетчеру исключительных ситуаций.

9. Диспетчер возвращает управление процедуре B, и выполнение процедуры B продолжается с команды, следующей за той, которая дала ошибку программы системного обслуживания.

10.5.1. Развертывание стека вызовов

Третий путь, который может выбрать обработчик кода состояния, заключается в том, чтобы развернуть стек вызовов процедур. Операция развертывания сложна, и ее следует использовать только в том случае, когда требуется передать управление какой-либо предшествующей процедуре в последовательности вызовов. Более того, применение программы системного обслуживания $UNWIND требует, чтобы вызывающему ее обработчику кода состояния была известна последовательность вызовов и точный адрес, которому будет передаваться управление.

Программа системного обслуживания $UNWIND допускает два необязательных аргумента:

1) глубина, до которой происходит развертывание. Если глубина равна 1, то стек вызовов разворачивается до вызывателя процедуры, создавшей исключительную ситуацию. Если глубина равна 2, то стек вызовов разворачивается до вызывателя процедуры, создавшей исключительную ситуацию, и т.д. задавая глубину из массива механизма, обработчик может осуществить развертывание до той процедуры, которая организовала данный обработчик;

2) адрес точки, в которую необходимо передать управление, когда выполнится операция развертывания, то есть PC,

который должен заменить текущий PC в кадре вызова той процедуры, которая получит управление, когда все заданные кадры будут удалены из стека.

Если служебной программе ¤UNWIND не передано ни одного аргумента, то развертывание будет выполняться до вызывателя той процедуры, которая организовала обработчик кода состояния, сделавший вызов служебной программы ¤UNWIND. Управление передается адресу, заданному в PC возврата для этой процедуры. Такой случай развертывания принят по умолчанию и является обычным случаем.

Другой общий случай развертывания заключается в развертывании до процедуры, об'явившей данный обработчик. Это легко сделать, используя значение глубины из массива механизма исключительной ситуации (CHF¤L_MCH_DEPTH) в качестве аргумента глубины для ¤UNWIND.

Отсюда следует, что развертывание по умолчанию (не задана глубина) эквивалентно заданию CHF¤L_MCH_DEPTH плюс один. Однако, в некоторых случаях вложенных исключительных ситуаций это не так. Рекомендуется не задавать аргумент глубины в тех случаях, когда осуществляется развертывание до вызывателя подпрограммы, организовавшей обработчик кода состояния.

На рис. 43 показана ситуация развертывания и описаны некоторые возможные результаты.

Непосредственно после вызова программы ¤UNWIND стек находится в промежуточном состоянии. В общем случае, обработчики должны возвращать управление сразу после вызова

¤UNWIND.

Во время фактического развертывания стека вызовов программа развертывания проверяет каждый кадр в стеке на наличие об'явленного в нем обработчика кода состояния. Если обработчик был об'явлен, то программа развертывания вызывает обработчик, задавая в аргументе "имя кода состояния" сигнального массива значение состояния SS¤_UNWIND (указывающее на то, что данный стек вызовов развертывается). После возврата из обработчика данный кадр вызова удаляется из стека.

Таким образом обработчик HANDLER3 на рис. 43 может быть вызван второй раз - во время операции развертывания. HANDLER3 не предназначен для того, чтобы специальным образом интерпретировать код состояния SS¤_UNWIND; просто инструкция RET возвращает управление процедуре развертывания, которая не проверяет никаких кодов состояния.

Развертывание стека вызовов

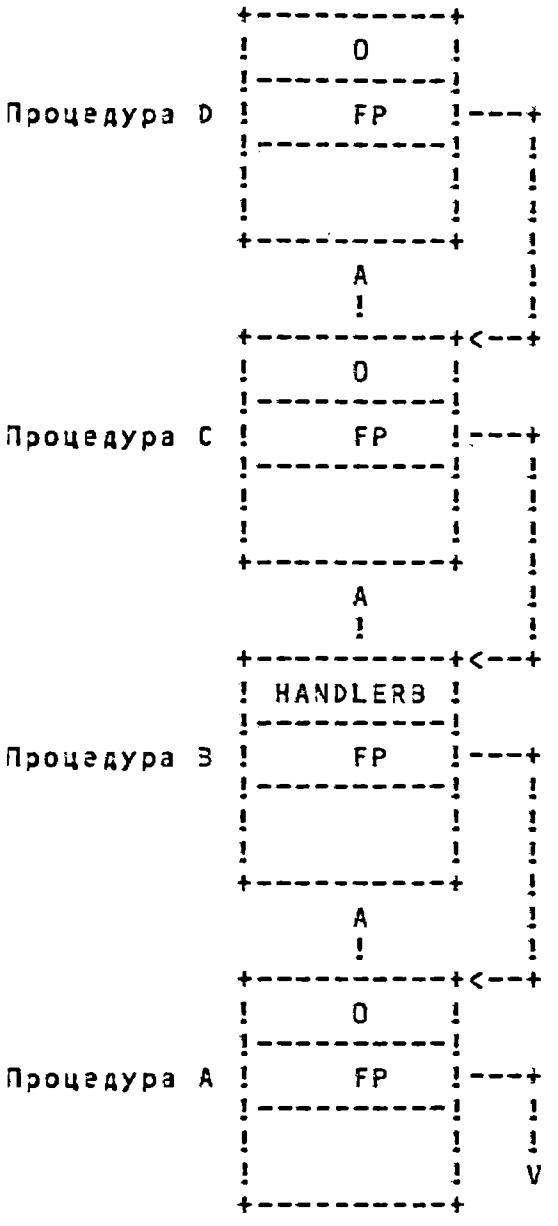


Рис. 43

Примечания:

1. Имеется показанный стек вызовов процедур. Предполагается, что для данного процесса не об'явлено никаких векторов исключительных ситуаций, и что исключительная ситуация возникла во время выполнения процедуры D.

2. Поскольку ни процедура D, ни процедура с не организовывали обработчик кода состояния, то управление получает HANDLERB.

3. Если HANDLERB вызывает программу системного обслуживания $UNWIND без указания аргументов, то кадры вызовов для процедур B, C и D удаляются из стека (вместе с кадром вызова самого обработчика HANDLERB), и управление передается процедуре A. Процедура a получает управление в точку, следующую за вызовом процедуры B.

4. Если HANDLERB вызывает программу системного обслуживания $UNWIND задавая глубину 2, то удаляются кадры вызовов для процедур C и D, и управление передается процедуре B.

10.6. Множественные исключительные ситуации

Возможно возникновение второй исключительной ситуации во время выполнения обработчика кода состояния или процедуры, которую вызвал обработчик. В этом случае, когда диспетчер исключительных ситуаций организует поиск обработчика кода состояния, он пропускает кадры, которые были просмотрены во время поиска первого обработчика.

Поиск второго обработчика заканчивается таким же образом, как и начальный поиск (подраздел 10.3).

Если второй активный обработчик кода состояния вызывает программу системного обслуживания $UNWIND, то глубина развертывания определяется в соответствии в теми же правилами, которые использует диспетчер исключительных ситуаций

при поиске в стеке; все кадры, просмотренные при поиске первого обработчика кода состояния, пропускаются.

С другой стороны, при возникновении исключительной ситуации всегда вводятся первичные и вторичные векторные обработчики.

Если какая-то исключительная ситуация возникла во время выполнения обработчика, об'явленного в первичном или вторичном векторе исключительной ситуации, то этот обработчик должен обрабатывать данное дополнительное состояние. Некорректность в этом случае может привести в рекурсивному циклу исключительной ситуации, в котором векторный обработчик будет повторно вызываться до тех пор, пока существует данный стек пользователя не будет исчерпан.

11. Служебные программы управления памятью

Программы управления памятью операционной системы мос вл. осуществляют отображение и управление взаимосвязями между физической памятью и виртуальным адресным пространством процессов. В большинстве случаев эти действия невидимы для пользователя и его программ. Однако в некоторых случаях пользователь может повысить эффективность программы при помощи явного управления использованием своей виртуальной памяти. Имеются следующие системные служебные программы управления памятью:

1) расширить область программы/управления (¤EXPREG);

2) сократить область программы/управления (¤CNTREG);

3) создать виртуальное адресное пространство (¤CRETVA);

4) удалить виртуальное адресное пространство (¤DELTVA);

5) создать и отобразить секцию (¤CRMPSC);

6) отобразить глобальную секцию (¤MGBLSC);

7) удалить глобальную секцию (¤DGBLSC);

8) обновить файл секции на диске (¤UPDSEC);

9) зафиксировать страницы в рабочем наборе (¤LKWSET);

10) освободить страницы в рабочем наборе (¤ULWSET);

11) отрегулировать размер рабочего набора (¤ADJWSL);

12) очистить рабочий набор (¤PURGWS);

13) зафиксировать страницы в памяти (¤LCKPAG);

14) освободить страницы в памяти (¤ULKPAG);

15) установить защиту страниц (¤SETPRT);

16) установить режим обмена процесса (¤SETSWM);

17) установить границы стека (¤SETSTK).

Служебные программы управления памятью позволяют регулировать размер виртуального и физического адресного пространства, доступного программе. Например, они дают возможность делать следующее:

1) увеличить или уменьшить виртуальное адресное пространство, доступное в области программы или в области управления процессов;

2) управлять размером рабочего набора процесса и осуществлять обмен страниц между физической памятью и устройством страничного обмена;

3) определять файлы на диске, содержащие данные или разделяемые образы, и отображать эти файлы на виртуальное адресное пространство процесса.

В данном разделе описаны служебные программы, обеспечивающие такие возможности. Однако, прежде чем обратиться к этим служебным программам, пользователь должен получить представление об организации памяти в СМ 1700 и о программах управления памятью (документ [16]).

11.1. Виртуальное адресное пространство

Виртуальное адресное пространство процесса подразделяется на две области:

1) область программы (PO), которая содержит образ, выполняемый в текущий момент времени;

2) область управления (P1), которая содержит информацию, организуемую операционной системой МОС ЭП для самого процесса. Она также содержит пользовательский стек, который расширяется по направлению к младшим адресам области управления.

На рис. 44 показана схема виртуальной памяти процесса. Начальный размер виртуального адресного пространства процесса зависит от размера выполняемого образа.

Для того, чтобы упростить защиту памяти и отображение, виртуальное адресное пространство подразделяется на 512-байтовые блоки, называемые страницами. Используя служебные программы управления памятью, процесс может добавить указанное количество страниц к концу области программы или области управления. Добавление страниц к области программы обеспечивает процесс дополнительным пространством для выполнения образа, например, для динамического создания таблиц или областей памяти. Добавление страниц к области управления увеличивает размер пользовательского стека. При обращении к новым страницам стек автоматически расширяется. (пользовательский стек можно также расширить при компоновке образа при помощи указания STACK=параметр в файле параметров компоновщика).

Максимальный размер, до которого процесс может увеличивать свое адресное пространство, определяется параметром VIRTUALPAGECNT обслуживающей программы SYSGEN.

11.2. Увеличение и уменьшение виртуального адресного пространства

Программа системного обслуживания ¤EXPREG добавляет страницы к концу либо области программы, либо области управления, и, если указано, возвращает диапазон виртуальных адресов новых страниц. Например, если желательно добавить к области программы процесса четыре страницы, то следует написать вызов программы системного обслуживания ¤EXPREG как показано в примере 1.

Пример 1.

```
BEGSPACE:
        .BLKL   2               два длинных слова для
                                хранения адресов начала и
                                конца новых страниц

        .
        .
        .
        ¤EXPREG_S -             получить 4 страницы
                PAGCNT=#4, -
                RETADR=BEGSPACE, -
                REGION=#0
```

В аргументе REGION передается значение 0, которое указывает на то, что страницы необходимо добавлять к области программы. Для того, чтобы добавить такое же количество страниц к области управления, следует задавать REGION=#1. Аргумент REGION в служебной программе ¤EXPREG необязателен; если он не указан, то страницы добавляются или удаляются по

умолчанию из области программы.

Рис. 44

Служебная программа ¤EXPREG может добавлять страницы только к концу области. Если требуется добавить страницы не к концу области, то можно воспользоваться программой системного обслуживания ¤CRETVA. И наоборот, если требуется удалить страницы, созданные либо программой ¤EXPREG, либо программой ¤CRETVA, можно использовать программу системного обслуживания ¤DELTVA. Например, если служебная программа ¤EXPREG дважды использовалась для добавления страниц, и если теперь требуется удалить первый диапазон страниц, оставляя второй, то следует использовать программу систем-

ного обслуживания ¤DELTVA, как показано в примере 2.

Пример 2.

```
BEGSPACEA:
        .BLKL   2               начало и конец первой об-
                                ласти
BEGSPACEB:
        .BLKL   2               начало и конец второй об-
                                ласти


        ¤EXPREG_S -             четыре страницы
                PAGCNT=#4, -
                RETADR=BEGSPACEA, -
                REGION=#0
        BSBW    ERROR


        ¤EXPREG_S -             еще три
                PAGCNT=#3, -
                RETADR=BEGSPACEB, -
                REGION=#0
        BSBW    ERROR
```

```
¤DELTVA_S -                удалить первые 4 страницы
        INADR=BEGSRACEA
BSBW    ERROR
```

В примере 2 первый вызов программы ¤EXPREG добавляет четыре страницы к области программы; виртуальные адреса созданных страниц возвращаются в массив из двух длинных слов BEGSPACEA. Второй вызов добавляет три страницы и возвращает адреса BEGSPACEB. Вызов программы ¤DELTVA удаляет первые четыре из добавленных страниц.

Примечание. Не следует использовать программы системного обслуживания ¤CNTREG или ¤CRETVA в сочетании с другими написанными пользователем процедурами. Эти программы системного обслуживания не имеют средств для того, чтобы сообщать изменение в виртуальном адресном пространстве другим программам. Рекомендуется использовать для получения памяти (там, где это проходит) либо ¤EXPREG, либо программу из исполнительной библиотеки LIB¤GET_VM (см. Документы [9]-[13]) (распределить виртуальную память). При использовании программы ¤DELTVA пользователь должен обеспечивать, чтобы удалялись только те страницы, которые он специально создавал.

11.2.1. Массивы входных адресов и массивы возвращаемых адресов

Программа системного обслуживания $EXPREG добавляет страницы к области в обычном направлении расширения данной области. Массив возвращаемых адресов, если он задан, указывает порядок, в котором добавлялись страницы:

1) если расширяется область программы, то начальный виртуальный адрес меньше чем конечный виртуальный адрес;

2) если расширяется область управления, то начальный виртуальный адрес больше, чем конечный виртуальный адрес.

Сокращение, выполняемое программой системного обслуживания $CNTREG, осуществляется для области программы в направлении от старших адресов к младшим, а для области управления - в направлении от младших адресов к старшим.

Возвращаемые адреса указывают на первый байт первой добавляемой или удаляемой страницы и на последний байт последней добавляемой страницы.

Если для системных служебных программ $CRETVA или $DELTVA указаны массивы входных адресов, то эти программы добавляют или удаляют страницы, начиная с адреса, заданного в первом длинном слове, и заканчивая адресом, заданным во втором длинном слове.

Порядок, в котором страницы добавляются или удаляются, не должен обязательно совпадать с направлением расширения, принятым для данной области. Более того, поскольку эти служебные программы добавляют или удаляют только целые страницы, они игнорируют младшие 9 битов заданных виртуальных

адресов (младшие 9 битов содержат смещение в байтах внутри страницы). Возвращаемые виртуальные адреса показывают смещение в байтах.

В табл. 38 приведены примеры виртуальных адресов, которые могут быть заданы как вход программ ¤CRETVA или ¤DELTVA и показаны массивы возвращаемых адресов, если все страницы были успешно добавлены или удалены.

Таблица 38

Входной массив		Область	Выходной массив		Количество страниц
Начало	Конец		Начало	Конец	
1010	1610	P0	1000	17FF	4
1450	1451	P0	1400	15FF	1
1200	1000	P0	1000	13FF	2
1450	1450	P0	1400	15FF	1
7FFEC010	7FFEC010	P1	7FFEC1FF	7FFEC000	1
7FFEC010	7FFEBCA0	P1	7FFEC1FF	7FFEBC00	3

Если входные виртуальные адреса одинаковые, как в четвертой и пятой строке табл. 38, то добавляется или удаляется одна страница. Массив возвращаемых адресов показывает, что страница была добавлена или удалена в соответствии с направлением расширения, принятым для данной области.

11.3. Владение страницами и защита страниц

Каждая страница в виртуальном адресном пространстве процесса принадлежит тому режиму доступа, который создал данную страницу. Например, страницы в области программы, с самого начала обеспечиваемые для выполнения образа, принадлежат пользовательскому режиму. Страницы, которые образ создает динамически, также принадлежат пользовательскому режиму. Страницы в области управления, за исключением страниц, содержащих пользовательский стек, обычно принадлежат более привилегированным режимам.

Только режим доступа владельца или более привилегированный режим может удалить страницу или как-то иначе воздействовать на нее. Владелец может также при помощи кода защиты указать тип доступа, который разрешается для каждого режима доступа.

Программа системного обслуживания $SETPRT (установить защиту страниц) изменяет защиту, присвоенную странице или группе страниц. Защита выражается кодом, который указывает специальный тип доступа (доступ запрещен, только чтение или чтение-запись) для каждого из четырех режимов доступа (режим ядра, режим управления, супервизорный или пользовательский режим). Защиту страниц может изменять только режим доступа владельца или более привилегированный режим доступа.

При попытке образа получить доступ к странице, которая защищена от предпринятого доступа, возникает аппаратная исключительная ситуация, вызванная нарушением доступа. Если

образ вызывает программу системного обслуживания, то она для страниц, которые должны использоваться, проверяет, не возникает ли нарушение доступа при попытке образа считать или записать одну из этих страниц. Если нарушение доступа возникает, то служебная программа осуществляет выход с кодом состояния SS¤_ACCVIO.

Поскольку служебные программы управления памятью добавляют, удаляют или модифицируют одновременно только одну страницу, то до того, как обнаружится нарушение доступа, могут быть успешно изменены одна или несколько страниц. Если в вызове служебной программы указан аргумент RETADR (возвращаемый адрес), то служебная программа возвращает адреса страниц, измененных (добавленных, удаленных или модифицированных) до появления ошибки. Если нет таких страниц, то есть нарушение доступа возникло на первой заданной странице, то служебная программа возвращает в оба длинных слова массива возвращаемых адресов значение "-1".

Если аргумент RETADR (возвращаемый адрес) не задан, то информация не возвращается.

11.4. Обмен страниц рабочего набора

Когда процесс выполняет образ, некоторое подмножество его страниц постоянно находится в физической памяти; эти страницы называются рабочим набором процесса. Рабочий набор включает страницы как области программы, так и области управления.

При обращении образа к странице, которой нет в памяти, возникает страничный отказ, и данная страница переносится в память, замещая существующую в рабочем наборе. Если страница, которую собираются заменить, модифицировалась во время выполнения образа, то она перезаписывается в файл обмена страниц на диск. Когда эта страница снова понадобится, она переносится в память, опять замещая текущую страницу в рабочем наборе. Такая передача страниц между физической и внешней памятью называется страничным обменом.

Однако, если программа очень большая или, если высока частота обмена страниц в интенсивно используемом программном образе, то накладные расходы, связанные со страничным обменом, снижают эффективность программы. Некоторые программы системного обслуживания позволяют процессу в пределах возможного разрешить эти потенциальные проблемы:

1) программа системного обслуживания $ADJWSL (отрегулировать размер рабочего набора) увеличивает или уменьшает максимальное количество страниц, которое процесс может иметь в своем рабочем наборе;

2) программа системного обслуживания $PURGWS удаляет одну или несколько страниц из рабочего набора;

3) программа системного обслуживания $LKWSET делает одну или несколько страниц в рабочем наборе запрещенными для страничного обмена.

Начальный размер рабочего набора процесса определяется принятой по умолчанию квотой рабочего набора процесса (WSDEFAULT). Поскольку некоторые программы могут требовать

больше памяти, чем другие, то программа может вызвать программу системного обслуживания ¤ADJWSL для того, чтобы динамически увеличить размер рабочего набора процесса. Если дополнительные страницы в рабочем наборе больше не требуются, программа может вызвать программу системного обслуживания ¤ADJWSL для того, чтобы уменьшить размер рабочего набора. Для того, чтобы удалить неиспользуемые больше страницы в рабочем наборе, можно также вызвать программу системного обслуживания ¤PURGWS.

При некоторых обстоятельствах может быть необходимым, чтобы определенные страницы никогда не обменивались. В этом случае образ может захватить эти страницы в рабочем наборе при помощи программы системного обслуживания ¤LKWSET. Пока данный рабочий набор процесса будет находиться в памяти, эти страницы не будут выгружаться, если только они не будут явно освобождены при помощи программы системного обслуживания ¤ULWSET.

11.5. Обмен процессов

Операционная система МОС ВП занимается балансированием нужд всех процессов, выполняемых в текущий момент времени, обеспечивая каждый процесс требуемыми ему системными ресурсами по мере возникновения потребности в них. Программы управления памятью уравновешивают потребность процессов в памяти. Сумма рабочих наборов всех процессов находящихся на текущий момент в физической памяти, называется балансным набором.

Когда процесс, рабочий набор которого находится в памяти, становится неактивным, например, переходит в состояние спячки или ожидания завершения операции ввода-вывода, этот рабочий набор, целиком или частично, может быть удален из памяти, чтобы освободилось пространство для рабочего набора другого процесса, который начнет выполняться.

Рабочий набор можно удалить двумя способами:

1) частично - так называемой подстройкой планировщика памяти. Страницы удаляются из рабочего набора процесса так, что количество страниц в рабочем наборе уменьшается, но никакие страницы не выгружаются из рабочего набора;

2) целиком - называется обменом. Все страницы из рабочего набора выгружаются.

Когда процесс выгружается из балансного набора, перезаписываются все страницы его рабочего набора (как модифицированные, так и немодифицированные), включая и те страницы, которые были захвачены в рабочем наборе.

Для привилегированного процесса существует возможность захватить в балансном наборе весь свой рабочий набор целиком. Чтобы захватить самого себя в балансном наборе, процесс вызывает программу системного обслуживания ¤SETSWM следующим образом.

Пример.

```
¤SETSWM_S        SWPFLG=#1
```

Такой вызов программы ¤SETSWM запрещает режим обмена процесса. Кроме того, режим обмена можно запретить при

помощи установки соответствующего бита в аргументе STSFLG программы системного обслуживания ¤CREPRC. Однако, для того, чтобы изменить режим обмена процесса, пользователь должен иметь привилегию PSWAPM.

Процесс может также захватить страницы в памяти при помощи программы системного обслуживания ¤LCKPAG. Если страница захвачена в памяти данной служебной программой, то она будет оставаться в памяти даже в том случае, если остальная часть рабочего набора выгружена из балансного набора. Данная программа системного обслуживания может быть полезной в определенных условиях, например для программ, которые выполняют операции ввода-вывода на медленные устройства или графические устройства.

Страницы, захваченные в памяти, можно освободить при помощи программы системного обслуживания ¤ULKPAG. Для вызова программы ¤LCKPAG или ¤ULKPAG требуется пользовательская привилегия PSWAPM.

11.6. Секции

Секция - это файл на диске или часть файла на диске, содержащая данные или инструкции, которые можно перенести в память и сделать доступными процессу для обработки и выполнения. Кроме того, секцией может быть одна или несколько последовательных физических страниц в физической памяти или в пространстве ввода-вывода (п. 11.6.14).

Секции бывают либо личные, либо глобальные (разделяемые):

1) личные секции доступны только тому процессу, который создал их. Процесс может определить дисковый файл данных как секцию, отобразить его в свое виртуальное адресное пространство и обрабатывать;

2) глобальные секции могут быть доступны нескольким процессам. В физической памяти находится одна копия глобальной секции, и каждый процесс, использующий ее, обращается к одной и той же копии. Глобальная секция может содержать разделяемые коды или данные, которые могут читаться, либо читаться и записываться несколькими процессами. Глобальные секции бывают либо временными, либо постоянными, и их можно определить как используемые внутри группы или как используемые в системе. Глобальные секции могут быть отображением файла на диске, либо созданными как глобальная секция физических страниц.

Если во время выполнения образа модифицированные страницы из секций дискового файла, допускающих запись, выгружаются из памяти, они перезаписываются обратно в секцию файла, а не в файл обмена страниц, как это обычно делается при работе с файлами. (однако, секция со ссылкой на копию не перезаписывается в секцию файла).

Использование секций дисковых файлов включает две различные операции:

1) создание секции определяет дисковый файл как секцию и информирует операционную систему МОС ВП о том, какие части файла содержит данная секция;

2) отображение секции делает секцию доступной процессу

и устанавливает соответствие между виртуальными блоками в файле и заданными адресами в виртуальном адресном пространстве процесса.

Программа системного обслуживания ¤CRMPSC создает и/или отображает личную или глобальную секцию. Поскольку личная секция используется только одним процессом, то создание и отображение выполняется одновременно. В случае глобальной секции один процесс может создать постоянную глобальную секцию и не отображать ее, отображение могут сделать другие процессы. Глобальную секцию можно также создать и отобразить в одной операции.

11.6.1. Создание секций

Для создания секций дисковых файлов требуются следующие шаги:

1) открытие или создание дискового файла, содержащего секцию;

2) определение того, какие виртуальные блоки в файле составляют секцию;

3) определение характеристик секции.

11.6.2. Открытие дискового файла

Прежде, чем файл можно будет использовать как секцию, его необходимо открыть при помощи системы управления данными СУД-32. Пример 1 показывает блок доступа к файлу ¤FAB и макрокоманду ¤OPEN, используемую для открытия файла, а так-

же определение канала для программы системного обслуживания ¤CRMPSC, необходимые для чтения существующего файла.

Пример 1.

```
SECFAB: ¤FAB    FNM=<SECTION.TST>,   блок доступа к файлу
                FOP=UFO
                RVT= -1


        ¤OPEN   FAB=SECFAB
        ¤CRMPSC_S -
                CHAN=SECFAB+FAB¤L_SVT,...
```

Параметр файла возможностей (FOP) показывает, что файл необходимо открыть для пользовательского ввода-вывода, то есть эта возможность требует, чтобы был назначен канал, используя режим доступа вызывающей программы. СУД-32 возвращает номер канала, которому доступен данный файл; этот номер канала задается как входной параметр служебной программы ¤CRMPSC (аргумент CHAN (канал)). Этот номер канала можно использовать для многократных операций создания и отображения секций.

Файл может быть временным, создаваемым на то время, пока он используется как секция. В этом случае для открытия файла следует использовать макрокоманду ¤CREATE.

Макрокоманду ¤CREATE можно использовать также для открытия существующего файла. Если файла нет, он будет создан. Пример 2 показывает поля в блоке доступа к файлу (FAB),

требуемые для условного создания файла.

Пример 2.

```
GBLFAB: ¤FAB    FNM=<GLOBAL.TST>, -
                ALQ=4, -
                FAC=PUT, -
                FOP=<UFO,CIF,CBT>, -
                SHR=<PUT,UPI>
```

```
        ¤CREATE FAB=GBLFAB
```

Когда вызывается макрокоманда ¤CREATE, она создает файл GLOBAL.TST, если он еще не существует. Возможность cbt (пытаться сделать непрерывным) запрашивает, чтобы файл был по возможности непрерывным. Хотя требование непрерывности секционных файлов не является обязательным, его выполнение может дать улучшение производительности.

11.6.3. Определение частей секции

Сразу после успешного открытия файла программа системного обслуживания ¤CRMPSC может создать секцию из всего файла или только из некоторых частей его. Части файла, которые составляют секцию, определяются аргументами программы ¤CRMPSC:

1) PAGCNT (счетчик страниц). Этот аргумент обязательный. Он задает количество отображаемых виртуальных блоков. Данные блоки соответствуют страницам секции;

2) VBN (номер виртуального блока). Этот аргумент определяет номер виртуального блока в файле, с которого начинается секция. Он не является обязательным. Если данный аргумент не задан, то передается значение 1 (началом секции является первый виртуальный блок в файле). Если задан номер отображаемой физической страницы, то аргумент VBN задает начальный номер физической страницы.

11.6.4. Определение характеристик секции

Аргумент FLAGS программы системного обслуживания $CRMPSC определяет следующие характеристики секции:

1) является ли данная секция личной или глобальной (по умолчанию создается личная секция);

2) как должны обрабатываться страницы секции, когда они копируются в физическую память или, когда процесс обращается к ним. Страницы в секции могут быть созданы:

- для чтения/записи или только чтения;
- как загружаемые обнуленными страницы или как страницы, со ссылкой на копию, в зависимости от того, как процесс собирается использовать эту секцию и содержит ли файл какие-либо данные (п. 11.6.9);

3) будет ли секция отображать дисковый файл либо заданные физические страницы (см. п. 11.6.14).

В табл. 39 показаны те биты флага, которые нужно устанавливать для задания характеристик.

Таблица 39

Биты флага для задания характеристик секции

Биты флага	Создаваемая секция				
	Личная	Глобальная	PFN Личная	PFN Глобальная	Разделяемая память
SEC$M_GBL	0	1	0	1	1
SEC$M_CRF	Необязат	Необязат	0	0	0
SEC$M_DZRO	Необязат	Необязат	0	0	Необязат
SEC$M_WRT	Необязат	Необязат	Необязат	Необязат	Необязат
SEC$M_PERM	Не исп.	Необязат	Необязат	1	1
SEC$M_SYSGBL	Не исп.	Необязат	Не исп.	Необязат	Необязат
SEC$M_PFNMAP	0	0	1	1	0
SEC$M_EXPREG	Необязат	Необязат	Необязат	Необязат	Необязат
SEC$M_PAGFIL	0	1	0	0	0

При задании характеристик секции необходимо выдерживать следующие ограничения:

1) глобальные секции не могут быть загружаемыми обнуленными, и со ссылкой на копию;

2) загружаемые обнуленными секции должны быть перезаписываемыми;

3) не допускаются личные секции разделяемой памяти.

11.6.5. Определение характеристик глобальной секции

Если секция является глобальной, то ей должно быть присвоено символьное имя (аргумент GSDNAM), чтобы другие процессы, отображая эту секцию, могли идентифицировать ее (подпункт 11.6.5.1.).

Аргумент FLAGS здает тип глобальной секции.

Глобальные секции могут быть:

1) групповыми временными (по умолчанию);

2) групповыми постоянными;

3) системными временными;

4) системными постоянными.

Групповые глобальные секции могут разделять только процессы, которые выполняются с тем же номером группы. Имя групповой глобальной секции неявно определяется номером группы процесса, создавшего эту секцию. Если секцию отображают другие процессы, то их номера группы должны совпадать.

Временная глобальная секция автоматически удаляется, когда ни один процесс не отображает ее, тогда как постоянная глобальная секция продолжает существовать даже в этом случае. Постоянную глобальную секцию можно пометить для удаления при помощи программы системного обслуживания $DGBLSC.

Для создания постоянной групповой глобальной секции или системной глобальной секции (временной или постоянной) требуются пользовательские привилегии PRMGBL и SYSGBL соответственно.

Системная глобальная секция доступна всем процессам в операционной системе МОС ВП.

По желанию процесс, создающий глобальную секцию, может задать маску защиты (аргумент PROT), запрещающую некоторые или все типы доступа (запись, чтение, выполнение, удаление) другим процессам.

11.6.5.1. Имя глобальной секции

Аргумент GSDNAM задает дескриптор, указывающий на символьную строку следующего формата:

[имя-разделяемой-памяти:] имя-глобальной-секции

где имя разделяемой памяти -

указывает, что создаваемая, отображаемая или удаляемая глобальная секция находится в пределах поименованной памяти, которую разделяют многие процессоры. Имя этой памяти задается во время генерации операционной системы МОС ВП. Например, строка SHRMEM¤1:GSDATA идентифицирует глобальную секцию с именем GSDATA, расположенную в разделяемой памяти с именем SHRMEM¤1.

Если имя разделяемой памяти не задано и осуществляется отображение или удаление данной секции, то операционная система МОС ВП пытается найти заданную глобальную секцию сначала в локальной памяти, а затем в блоках разделяемой памяти (в том порядке, как они соединены);

имя глобальной секции -

это имя присваивается глобальной секции. Можно выбирать любое допустимое имя от 1 до 43 символов. Однако

все процессы, отображающие одну и ту же глобальную секцию, должны указывать одинаковое имя.

По желанию для глобальной секции, которая расположена в памяти, разделяемой многими процессорами, можно задавать и имя-разделяемой-памяти, и имя-глобальной-секции. Однако, если пользователь хочет использовать существующие программы без повторной компиляции или компоновки, или чтобы программа работала независимо от того, находится секция в локальной или разделяемой памяти, он может задавать только имя-глобальной-секции и обеспечить, чтобы операционная система MOC ЭП оттранслировала его для получения полной спецификации. Операционная система MOC ЭП пытается выполнить трансляцию логического имени для строки, заданной аргументом GSDNAM, следующим образом:

1) осуществляется поиск двоеточия в строке имени. Если двоеточие найдено, то считается, что глобальная секция должна находиться в разделяемой памяти, и трансляция продолжается следующим образом:

- часть строки имени справа от двоеточия помещается в буфер имени-глобальной-секции. Часть строки имени слева от двоеточия становится новой текущей строкой имени;
- к текущей строке имени добавляется префикс GBL¤ и результат подвергается трансляции логического имени;
- если результат содержит логическое имя, то шаги 1 и 2 повторяются до тех пор, пока трансляция не закончится ошибкой или пока количество выполненных транс-

ляций не превысит числа, заданного параметром системной генерации LNM$C_MAXDEPTH;

- если текущая строка имени не может быть оттранслирована, то префикс GBL$ убирается из нее. Это имя становится именем разделяемой памяти. Именем-глобальной-секции является текущая строка, содержащаяся в буфере имени-глобальной-секции;

2) если глобальная секция размещается в локальной памяти, то трансляция осуществляется следующим образом:

- к текущей строке имени добавляется префикс GBL$ и результат подвергается трансляции логического имени;
- если результат является логическим именем, то шаг 1 повторяется до тех пор, пока трансляция не закончится ошибкой или пока количество выполненных трансляций не превысит числа, заданного параметром системной генерации LNM$C_MAXDEPTH;
- если текущая строка имени не может быть оттранслирована, то префикс GBL$ убирается из нее. Эта текущая строка является именем-глобальной-секции.

Пример 1.

```
$DEFINE GBL$GSDATA      SHRMEM$1:GSDATA
```

Пример 2.

```
NAMEDESC:
        .ASCID  /GSDATA/            дескриптор для логического имени секции
```

```
$CRMPSC_S -
        GSDNAM=NAMEDESC,...
```

Трансляция логического имени осуществляется следующим образом:

1) к SHRMEM$1 добавляется префикс GBL$;

2) GBL$GSDATA транслируется в SHRMEM$1:GSDATA. (для GBL$SHRMEM$1 дальнейшая трансляция не будет успешной. Когда трансляция логического имени заканчивается по ошибке, то данная строка передается служебной программе).

Для метода трансляции логического имени, описанного в данном подпункте, существует три исключительных ситуации:

1) если строка имени начинается знаком подчеркивания (_), операционная система МОС ВП убирает знак подчеркивания и считает результирующую строку фактическим именем (то есть, дальнейшая трансляция не выполняется);

2) если строка имени является результатом трансляции логического имени, то проверяется, имеет ли данная строка имени атрибут "TERMINAL". Если строка имени помечена атрибутом "TERMINAL", то операционная система МОС ВП считает результирующую строку фактическим именем (то есть, дальнейшая трансляция не выполняется);

3) если глобальная секция имеет имя в формате имя_NNN, то операционная система МОС ВП сначала удаляет знак подчеркивания и цифры (NNN), затем транслирует полученое имя в соответствии с той последовательностью, которая изложена в данном подпункте, и после этого снова добавляет знак под-

черкивания и цифры.

11.6.6. Отображение секций

При вызове программы системного обслуживания $CRMPSC для создания и/или отображения секции, необходимо передать программе диапазон виртуальных адресов (аргумент INADR), на который требуется отобразить секцию.

Если страницы, на которые требуется отобразить секцию, известны, то их адреса помещают в массив из двух длинных слов. Например, чтобы отобразить личную секцию из 10 страниц на виртуальные страницы от 10 до 19 в области программы, требуется задать следующий массив входных адресов:

Пример 1.

```
MAPRANGE:
        .LONG   ^X1400          адрес (шестнадцатеричный)
                                страницы 10
        .LONG   ^X2300          адрес (шестнадцатеричный)
                                страницы 19
```

Однако, для того, чтобы обеспечить диапазон входных адресов, не обязательно знать явные адреса. Если пользователю необходимо отобразить секцию на первый доступный диапазон виртуальных адресов в области программы (P0) или в области управления (P1), он может задать бит флага SEC$M_EXPREG в аргументе FLAGS. В этом случае адреса, задаваемые в аргументе INADR, управляют тем, что служебная программа находит первый доступный диапазон либо в области программы, либо в области управления. Значение, заданное

либо принимаемое по умолчанию для аргумента PAGCNT, определяет количество отображаемых страниц. В примере 2 показана часть программы, которая используется для отображения секцию на текущий конец области программы.

Пример 2.

```
MAPRANGE:
        .LONG   ^X200           какой-нибудь адрес области программы (P0)
        .LONG   ^X200           какой-либо адрес P0 (может быть тот же самый)
RETRANGE:
        .BLKL   2               здесь возвращаемый диапазон адресов


        $CRMPSC_S -
                INADR=MAPRANGE, -
                RETADR=RETRENGE, -
                FLAGS=<SEC$M_EXPREG>,...
```

Задаваемые адреса необязательно должны быть в данный момент в виртуальном адресном пространстве процесса. Программа системного обслуживания $CRMPSC создает требуемое виртуальное адресное пространство во время отображения сек-

ции. Если задан аргумент RETADR (возвращаемый адрес), то служебная программа возвращает диапазон фактически отображенных адресов.

Сразу после того, как секция успешно отображена, образ может обращаться к страницам, используя одно из следующих средств:

1) базовый регистр или указатель и заранее определенные имена символических смещений;

2) метки, определяющие смещения абсолютной программной секции или структуры.

В примере 3 показана часть программы, используемая для создания и отображения секции процесса.

Пример 3.

```
SECFAB: $FAB    FNM=<SECTION.TST>, -
                FOP=UFO, -
                FAC=PUT, -
                SHR=<GET,PUT>

MAPRANGE:
        .LONG   ^X1400          первая страница
        .LONG   ^X2300          последняя страница
RETRANGE:
        .BLKL   1               первая отображенная
                                страница
ENDRANGE:
        .BLKL   1               последняя отображен-
                                ная страница
```

```
¤OPEN   FAB=SECFAB          открыть файл секции
        BSBW    ERROR
¤CRMPSC_S -
        INADR=MAPRANGE,-; массив входных адресов
        RETADR=RETRANGE,-   выходной массив
        PAGCNT=#4, -        отобразить 4 страницы
        FLAGS=#SEC¤M_WRT,-; секция доступна для
                            чтения/записи
        CHAN=SECFAB+FAB¤L_STV       номер канала
        BSBW    ERROR
        MOVL    RETRANGE,R6     указать на начало
                                секции
```

Примечания:

1. Макрокоманда ¤OPEN открывает файл секции, определенный в блоке доступа к файлу SECFAB. (параметр FOP в макрокоманде ¤FAB должен задавать возможность UFO (пользовательский ввод-вывод)).

2. Программа системного обслуживания ¤CRMPSC использует адреса, заданные в MAPRANGE, для того, чтобы указать входной диапазон адресов, на который будет отображаться данная секция. Аргумент PAGCNT запрашивает отображение только четырех страниц.

3. Аргумент FLAGS требует, чтобы страницы в секции имели доступ "чтение/запись". Символические определения флагов для этого аргумента определены в макрокоманде

¤SECDEF. Поле доступа к файлу (параметр FAC) в блоке FAB также указывает, что файл должен быть открыт для записи.

4. После выполнения прогаммы ¤CRMPSC адреса четырех отображенных страниц возвращаются в массив выходных адресов RETRANGE. В регистр R6 помещается адрес начала секции, который служит указателем секции.

11.6.7. Отображение глобальных секций

Процесс, создающий глобальную секцию, может при создании и отобразить ее. Потом ее могут отобразить другие процессы, вызывая программу системного обслуживания ¤MGBLSC (отобразить глобальную секцию).

Когда процесс отображает глобальную секцию, он должен указать имя глобальной секции, присвоенное данной секции при ее создании, является ли она групповой или системной глобальной секцией, и тип доступа: только чтение или чтение/запись. Кроме того, процесс может указать:

1) идентификацию версии (аргумент IDENT), задающую номер версии глобальной секции (если существует несколько версий), а также, доступны ли процессу более поздние версии;

2) относительный номер страницы (аргумент RELPAG), задающий номер страницы относительно начала секции, с которого начинается отображение секции. В этом случае процессы могут использовать только части секции. Процесс может отобразить часть секции на конкретный диапазон адресов, а впоследствии отобразить на этот же диапазон виртуальных адресов

другую часть секции.

Для того, чтобы указать, что отображаемая глобальная секция размещается в физической памяти, которую разделяют многие процессоры, можно включить в символьную строку аргумента GSDNAM имя разделяемой памяти (см. подпункт 11.6.5.1.). Загружаемые обнуленными глобальные секции, которые находятся в памяти, разделяемой многими процессорами, должны отображаться во время их создания.

Взаимосвязанные процессы оба могут вызвать программу системного обслуживания $CRMPSC для создания и отображения одной и той же глобальной секции. Первый процесс, вызвавший служебную программу, фактически создает глобальную секцию. Последующие попытки создать и отобразить данную секцию приводят только к отображению секции для вызывающей программы. Возвращаемый успешный код состояния SS$_CREATED показывает, что при вызове программы системного обслуживания $CRMPSC секция еще не существовала. Если секция существовала, то возвращается код состояния SS$_NORMAL.

11.6.8. Глобальные секции файла страниц

Глобальные секции файла страниц используются для хранения временных данных в глобальной секции. Глобальная секция файла страниц - это секция виртуальной памяти, которая не отображается в файл. Эту секцию можно удалить, если все процессы закончили работу с ней. Общее количество глобальных страниц файла страниц в операционной системе МОС ВП ограничивается параметром GBLPAGFIL системной генерации.

Чтобы создать глобальную секцию файла страниц, пользователь должен установить биты флага SEC$M_GBL и SEC$M_PAGFIL в аргументе FLAGS программы системного обслуживания $CRMPSC. Номер канала (аргумент CHAN) должен быть равным нулю.

Нельзя одновременно устанавливать биты флага SEC$M_CRF и SEC$M_PAGFIL.

11.6.9. Страничный обмен секций

Когда образ, работающий в процессе, впервые обращается к странице, созданной во время отображения секции дискового файла, данная страница копируется в физическую память. Адрес страницы в виртуальном адресном пространстве процесса отображается на физическую страницу. Во время выполнения образа может осуществляться обычный обмен страниц. Однако страницы секции, если они выгружаются, не записываются в файл обмена страниц, как это обычно бывает при обмене. Вместо этого они, если были модифицированы, перезаписываются в файл секции на диск. Следующее обращение к странице приведет к страничному отказу, и страница будет загружена из файла секции.

Однако, если при создании секции страницы в ней были определены как загружаемые обнуленными, или как страницы со ссылкой на копию, то эти страницы обрабатываются следующим образом:

1) если при вызове программы $CRMPSC было указано, что страницы в секции должны рассматриваться как загружаемые по

запросу обнуленные страницы, то при создании этих страниц в физической памяти они инициализируются нулями. Загрузка обнуленных по запросу страниц обеспечивает удобный способ инициализации страниц файла в том случае, если создается новый файл как секция, или нужно полностью перезаписать существующий файл. Эти страницы снова выгружаются в файл секции;

2) при удалении виртуального адресного пространства все страницы, к которым не было обращений, записываются обратно в файл как нулевые. Это приводит к инициализации файла несмотря на то, что часть страниц была модифицирована;

3) если при вызове программы $CRMPSC было указано, что страницы в секции со ссылкой на копию, то каждый процесс, который отображает данную секцию, получает свою собственную копию секции по принципу загрузки страницы за страницей по мере обращения к ним. Эти страницы никогда не перезаписываются обратно в файл секции, но при необходимости они выгружаются в файл обмена страниц.

Если секция является глобальной, то несколько процессов могут отображать ее на одни и те же физические страницы. Если требуется эти страницы выгрузить или перезаписать в дисковый файл, определенный как секция, то такие операции выполняются только в том случае, когда страницы не находятся в рабочем наборе какого-либо процесса.

Пример.

```
Процесс ORION
FLGCLUSTER:                          дескриптор для имени
        .ASCID  /FLAG_CLUSTER/       кластера общих флагов
                                     событий
GLOBALSEC:                           дескриптор для имени гло-
        .ASCID  /GLOBAL_SECTION/;    бальной секции

FLGSET = 65                          номер флага, с которым
                                     нужно связаться и устано-
                                     вить его
GBLFAB: ¤FAB    FNM=<GLOBAL.TST>, -
                FOP=<UFO,CIF,CBT>, -
                ALQ=4, -
                FAC=PUT



        ¤ASCEFC_S -
                EFN=#FLGSET, -
                NAME=FLGCLUSTER
        BSBW    ERROR
        ¤CRMPSC_S -                  создать глобальную секцию
                GSDNAM=GLOBALSEC, -
                FLAGS=#SEC¤M_WRT!SEC¤M_GBL,...
        BSBW    ERROR
        ¤SETEF_S -                    установить общий флаг со
```

```
                EFN=#FLGSET         бытия
процесс CYGNUS
CLUSTER:
        .ASCID  /FLAG_CLUSTER/      дескриптор для имени
                                    кластера
SECTION:
        .ASCID  /GLOBAL_SECTION/    дескриптор имени секции
FLGSET  =  65



        ¤ASCEFC_S -
                EFN=¤FLGSET, -
                NAME=CLUSTER
        BSBW    ERROR
        ¤WAITFR_S
                EFN=#FLGSET
        BSBW    ERROR
        ¤MGBLSC_S -
                INADR=MAPRANGE, -
                RETADR=RETRANGE, -
                FLAGS=#SEC¤M_GBL,- ; глобальная секция
                GSDNAM=SECTION,-     имя секции
        BSBW    ERROR
```

Примечания:

1. Процессы ORION и CYGNUS принадлежат к одной группе. Каждый процесс сначала связывается с кластером общих флагов

событий, имеющим имя FLAG_CLUSTER, для того, чтобы использовать общие флаги событий для синхронизации своих обращений к секции.

2. Процесс ORION создает глобальную секцию с именем GLOBAL-SECTION, задавая флаги секций, которые указывают, что секция является глобальной (SEC$M_GBL) и что она имеет тип доступа "чтение/запись". Массивы входных и выходных адресов, аргументы параметра счетчика страниц и номера канала не показаны.

3. Процесс CYGNUS связывается с кластером общих флагов событий и ждет установки флага, определенного как FLGSET. Процесс ORION устанавливает этот флаг, когда он заканчивает создание секции. Для того, чтобы отобразить эту секцию, процесс CYGNUS указывает массивы входных и выходных адресов, флаг, определяющий, что секция является глобальной, и имя глобальной секции. В данном примере отображается такое же количество страниц, которое было указано при создании секции.

11.6.10. Секции чтения и записи данных

Секции, допускающие чтение и запись, обеспечивают процессу или взаимосвязанным процессам способ для разделения файлов данных в физической памяти.

Например, один процесс может создавать и отображать глобальную секцию, доступную для чтения и записи; другие процессы могут отображать ее только для чтения и обрабатывать данные, записанные первым процессом. Или два и более

процессов могут совместно писать в секцию. (в этом случае прикладные программы должны обеспечивать необходимую синхронизацию и защиту).

Если файл уже обновлен, процесс или процессы могут освободить (или отобразить обратно) секцию. Тогда модифицированные страницы перезаписываются в файл на диске, определенный как секция.

11.6.11. Освобождение и удаление секций

Процесс освобождает секцию, удаляя из своего виртуального адресного пространства те адреса, на которые отображалась данная секция. Если для получения адресов отображенных страниц была указана область возвращаемых адресов, то эту область адресов можно использовать как входной параметр программы системного обслуживания $DELTVA.

Пример 1.

```
$DELTVA_S        INARD=RETRANGE
```

Если процесс освобождает личную секцию, то она удаляется, т.е. удаляется вся управляющая информация, созданная операционной системой МОС ВП. Временная глобальная секция удаляется, когда ее освобождают все процессы, которые отображали эту секцию. Постоянные глобальные секции не удаляются до тех пор, пока они не будут специально помечены при помощи программы системного обслуживания $DGBLSC. В этом случае они будут удалены, когда ни один процесс не будет отображать их.

Удаление страниц, занятых секцией, не удаляет файл секции, а только отменяет связь процесса с данным файлом. Когда процесс удаляет страницы, на которые отображена секция, доступная для чтения и записи, и никакие другие процессы не отображают ее, то все модифицированные страницы записываются обратно в файл секции.

Если секция уже удалена, то присвоенный ей канал можно отсоединить. Процесс, создавший данную секцию, может отсоединить канал (при помощи программы системного обслуживания "отсоединить канал ввода-вывода").

Пример 2.

```
¤DASSGN_S       CHAN=G3LFAB+FAB¤L_STV
```

11.6.12. Перезапись секций

Поскольку обычно секции, доступные для чтения и записи, не обновляются на диске до тех пор, пока не выгружаются страницы, которые они занимают, или пока все процессы, обращающиеся к секции, не освободят ее, - процессу может потребоваться гарантия того, что все модифицируемые страницы успешно перезаписываются обратно в файл секции через регулярные интервалы времени.

Модифицированные страницы секции перезаписывает в файл на диске программа системного обслуживания ¤UPDSEC.

11.6.13. Секции образов

Глобальные секции могут содержать разделяемый код. Операционная система МОС ВП использует глобальные секции для реализации разделяемого кода следующим образом:

1) об'ектный модуль, который содержит код, подлежащий разделению, компонуется для создания разделяемого образа. Этот разделяемый образ сам по себе не является выполняемым. Он содержит ряд секций, называемых секциями образа;

2) пользователь компонует личные об'ектные модули с этим разделяемым образом для получения выполняемого образа. Никакие коды или данные из разделяемого образа не включаются в выполняемый образ;

3) для того, чтобы сделать секции образа доступными для разделения, системный администратор при помощи команды INSTALL создает из файла разделяемого образа постоянную секцию;

4) во время работы выполняемого образа операционная система МОС ВП автоматически отображает глобальные секции, созданные командой INSTALL, на виртуальное адресное пространство пользовательского процесса (документы [17] и [18]).

11.6.14. Секции физических страниц

Секция физических страниц - это одна или несколько смежных страниц физической памяти или пространство ввода-вывода, которое необходимо отобразить как секцию. Секции физических страниц можно использовать для отображе-

ния страницы ввода-вывода, что позволяет процессу читать регистры устройства. Процесс, отображенный на страницу ввода-вывода, может также подсоединяться к вектору прерываний устройства.

Секция физических страниц отличается от секции дискового файла тем, что она не связана ни с каким отдельным файлом на диске и не выгружается. Однако она похожа на секцию дискового файла во многих отношениях. Пользователь создает, отображает, определяет характеристики секции физических страниц в основном так же, как это делается для секции дискового файла.

Для создания секции физических страниц необходимо задать номер физической страницы при помощи установки бита SEC$M_PFNMAP в аргументе FLAGS служебной программы $CRMPSC. Аргумент VBN используется для создания первой отображаемой физической страницы, а не первого виртуального блока. Пользовательская привилегия PFNMAP необходима для создания или удаления секции физических страниц, но для отображения существующей секции она не нужна.

Поскольку данный тип секции не связан с файлом на диске, то аргументы RELPAG, CHAN и PFC служебной программы $CRMPSC не используются при создании или отображении физических страниц. По той же причине неприменимы установки битов SEC$M_CRF (ссылка на копию) и SEC$M_DZRO (загрузка обнуленных страниц).

При работе с секциями физических страниц нужно соблюдать осторожность. Если разрешается запись в секцию, то

каждый процесс, который туда пишет, делает это на свой риск. Могут возникнуть серьезные ошибки, если процесс записывает некорректные данные или ошибочно пишет на другую страницу, особенно если эта страница отображается также операционной системой МОС ВП или другим процессом. Таким образом, пользователь, имеющий привилегию PFNMAP, может нарушить или разрушить защиту операционной системы МОС ВП.

12. Служебные программы управления захватом ресурсов

Служебные программы управления захватом операционной системы МОС ВП позволяют взаимосвязанным процессам синхронизировать их доступ к разделяемым ресурсам. Это достигается путем обеспечения общей области данных, в которой процессы могут захватить указанный ресурс по имени. Все процессы, которые осуществляют доступ к ресурсам, должны использовать служебные программы управления захватом, иначе они не будут эффективными.

Для того, чтобы синхронизировать доступ к ресурсам, служебные программы управления захватом обеспечивают механизм очередей, что позволяет процессам ожидать в очереди, пока конкретный ресурс не станет доступным.

Системная программа $ENQ (поставить в очередь запрос на захват) используется для того, чтобы осуществлять запросы на захват, а системная служебная программа $DEQ (удалить из очереди запрос на захват) - для того, чтобы отменять запросы на захват. Программа системного обслуживания

¤GETLKI (получить информацию о захвате) применяется для получения информации о существующих захватах.

12.1. Концепции ресурсов и захватов

Ресурсом может быть все, что существует в операционной системе МОС ВП (например, файлы, структуры данных, базы данных и выполняемые программы). Если к одному и тому же ресурсу осуществляют доступ два или несколько процессов, то часто возникает необходимость управлять их доступом к данному ресурсу. Нежелательно, чтобы один процесс читал ресурс в то время, как другой процесс пишет новые данные. Записывающая программа может быстро испортить что-либо из того, что считывает читающая программа. Системные служебные программы управления захватом позволяют связать имя с ресурсом и запросить доступ к этому ресурсу. Режимы захвата разрешают процессам указать, как они хотят разделять доступ с другими процессами.

Процесс должен запрашивать доступ к ресурсу (запрашивать захват) при помощи системной служебной программы ¤ENQ. Для захватов системной служебной программе ¤ENQ требуется три аргумента:

1) имя ресурса. Служебные программы управления захватом используют имя ресурса для того, чтобы проверить, нет ли других запросов на захват, использующих это же имя;

2) режим захвата, соответствующий запрашиваемому захвату. Режим захвата определяет, как процесс будет разделять ресурс с другими процессами;

3) адрес блока состояния захвата. Блок состояния захвата получает код состояния выполнения захвата и идентификацию захвата. Идентификация захвата используется для обращения к запросу захвата после того, как он будет поставлен в очередь.

Служебные программы управления захватом сравнивают режим вновь запрашиваемых захватов с режимом захвата других захватов с тем же именем ресурса:

1) если нет других процессов, имеющих захват данного ресурса, то новый захват предоставляется;

2) если есть другой процесс, имеющий захват данного ресурса, и режим нового запроса совместим с существующим захватом, то новый захват предоставляется;

3) если другой процесс уже захватил ресурс, и режим нового запроса несовместим с режимом существующего захвата, то новый запрос ставится в очередь, где он ожидает, пока не станет доступным. Когда ресурс становится доступным, процесс извещают о том, что он может осуществлять доступ к данному ресурсу (захват предоставлен).

Кроме того, процесс может использовать системную служебную программу #ENQ для изменения режима захвата. Это называется преобразованием захвата.

12.1.1. Иерархичность захватов

Многие ресурсы можно подразделить на меньшие части. До тех пор, пока часть ресурса можно идентифицировать именем, эту часть можно захватить.

Рис. 45 иллюстрирует модель базы данных. База данных делится на файлы, которые в свою очередь подразделяются на записи. Далее записи делятся на элементы.

Модель базы данных

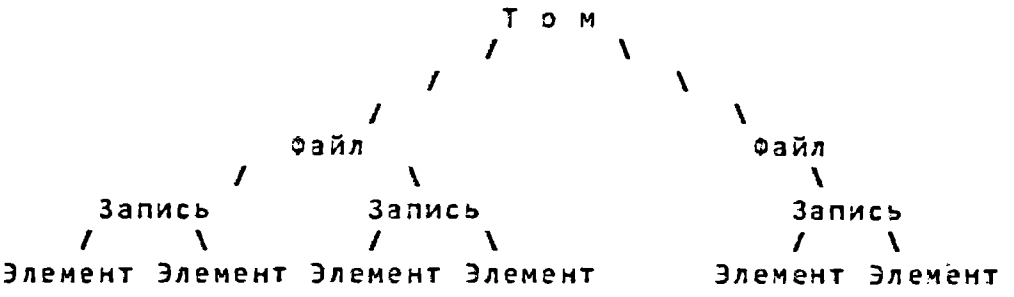


Рис. 45

Процессы, которые запрашивают захваты этой базы данных, должны решить, будут ли они захватывать всю базу данных, файл в базе данных, запись или единственный элемент. Захват всей базы данных считается захватом на верхнем уровне иерархии. Захват единственного элемента - захват на нижнем уровне иерархии.

12.1.2. Имена ресурсов

Системные служебные программы управления захватом обращаются к каждому ресурсу по имени, составленному из четырех частей:

1) имя, заданное вызывающей программой;

2) режим доступа вызывающей программы;

3) номер группы UIC вызывающей программы (если этот ресурс не является общесистемным);

4) идентификация процесса, породившего этот захват (не обязательна).

Для двух ресурсов, под которыми подразумевается один и тот же ресурс, эти четыре части должны быть идентичны.

Имя, заданное процессом, представляет ресурс, подлежащий захвату. Другие процессы, которым необходимо получить доступ к этому ресурсу, должны обращаться к нему при помощи такого же имени. Соответствие имени и ресурса устанавливается по соглашению между взаимосвязанными процессами.

Режим доступа определяется режимом доступа вызывающей программы, если в вызове системной служебной программы $ENQ не задан менее привилегированный режим (п. 2.1.3.).

Ресурсы могут быть группового значения или общесистемными. По умолчанию имена ресурсов определяются номером группы UIC вызывающего процесса. Общесистемные ресурсы определяются установкой бита флага в вызове системной служебной программы $ENQ. Для того, чтобы запросить общесистемные захваты из пользовательского или супервизорного режимов, необходимо иметь пользовательскую привилегию SYSLCK. Для захвата общесистемных ресурсов из режима управления или режима ядра никакие дополнительные привилегии не требуются.

Если ставится в очередь запрос на захват и указана идентификация порождающего захвата, то запрашиваемый захват становится подзахватом по отношению к указанному порождающему захвату. Однако, порождающий захват должен быть предоставлен, иначе запрос на захват не принимается. Таким

способом процесс может осуществлять захват ресурса с различной степенью иерархии.

12.1.3. Выбор режима захвата

Режим захвата определяет, может или нет данный ресурс быть разделен другими запросами захвата. В операционной системе МОС ВП определены шесть режимов захвата:

LCK¤K_NLMODE -

нулевой режим. Данный режим не предоставляет доступа к ресурсу. Нулевой режим обычно используется как индикатор потребности в ресурсе либо как средство занятия места для будущих преобразований захвата;

LCK¤K_CRMODE -

совместное чтение. Данный режим предоставляет доступ на чтение ресурса и допускает разделение ресурса с другими читающими процессами. Режим совместного чтения обычно используется в тех случаях, когда осуществляется дополнительный захват на нижнем уровне иерархии с подзахватами, или для чтения данных из ресурса "без защиты" (допускающего одновременную запись в ресурс);

LCK¤K_CWMODE -

совместная запись. Данный режим предоставляет доступ на запись в ресурс и допускает разделение ресурса с другими записывающими процессами. Режим совместной записи обычно используется для выполнения дополнительного захвата на нижнем уровне иерархии или для записи в ресурс "без защиты";

LCK¤K_PRMODE -

защищенное чтение. Данный режим предоставляет доступ на чтение ресурса и допускает разделение ресурса с другими читающими процессами. Записывающим процессам запрещается доступ к ресурсу. Это обычный "разделяемый захват";

LCK¤K_PWMODE -

защищенная запись. Данный режим предоставляет доступ на запись в ресурс и допускает разделение ресурса с захватами, имеющими режим совместного чтения. Это обычный "захват для обновления";

LCK¤K_EXMODE -

монопольный режим. Данный режим предоставляет доступ на запись в ресурс и запрещает разделение этого ресурса с любыми другими читающими или записывающими процессами. Это обычный "монопольный захват".

12.1.4. Уровни захвата и совместимость

Захваты, которые разрешают процессу разделять ресурс, называются захватами низкого уровня. Захваты, которые дают практически монопольный доступ к ресурсу, называются захватами высокого уровня. Захваты с нулевым режимом и режимом совместного чтения считаются захватами низкого уровня. Захваты с режимом защищенной записи и с монопольным режимом считаются захватами высокого уровня. По возрастанию уровня режимов доступа режимы захвата располагаются в следующем порядке: нулевой, совместное чтение, совместная запись,

защищенное чтение, защищенная запись и монопольный режим. Режим совместной записи и режим защищенного чтения считаются режимами одного уровня.

Разделяемые захваты должны иметь совместимые режимы захвата. Режимы захвата высших уровней менее совместимы с другими, чем режимы захвата низших уровней. В табл. 40 показана совместимость режимов захвата.

Таблица 40

Совместимость режимов захвата

Режим запрашиваемого захвата	Режим представленных на текущий момент времени захватов					
	NL	CR	CW	PR	PW	EX
NL	Да	Да	Да	Да	Да	Да
CR	Да	Да	Да	Да	Да	Нет
CW	Да	Да	Да	Нет	Нет	Нет
PR	Да	Да	Нет	Да	Нет	Нет
PW	Да	Да	Нет	Нет	Нет	Нет
EX	Да	Нет	Нет	Нет	Нет	Нет

Примечания:

Обозначение режимов захвата:

NL - нулевой захват.

CR - совместное чтение.

CW - совместная запись.

PR - защищенное чтение.

PW - защищенная запись.

EX - монопольный режим.

12.1.5. Очереди управления захватом

Запрос на захват может быть в одном из трех состояний:

GRANTED - запрос на захват удовлетворен;

WAITING - запрос на захват ожидает разрешения;

CONVERSION - запрос на захват удовлетворен для одного режима и ожидает разрешения для режима захвата более высокого уровня.

Каждому из трех состояний соответсвует очередь (рис. 46).

Три очереди захватов

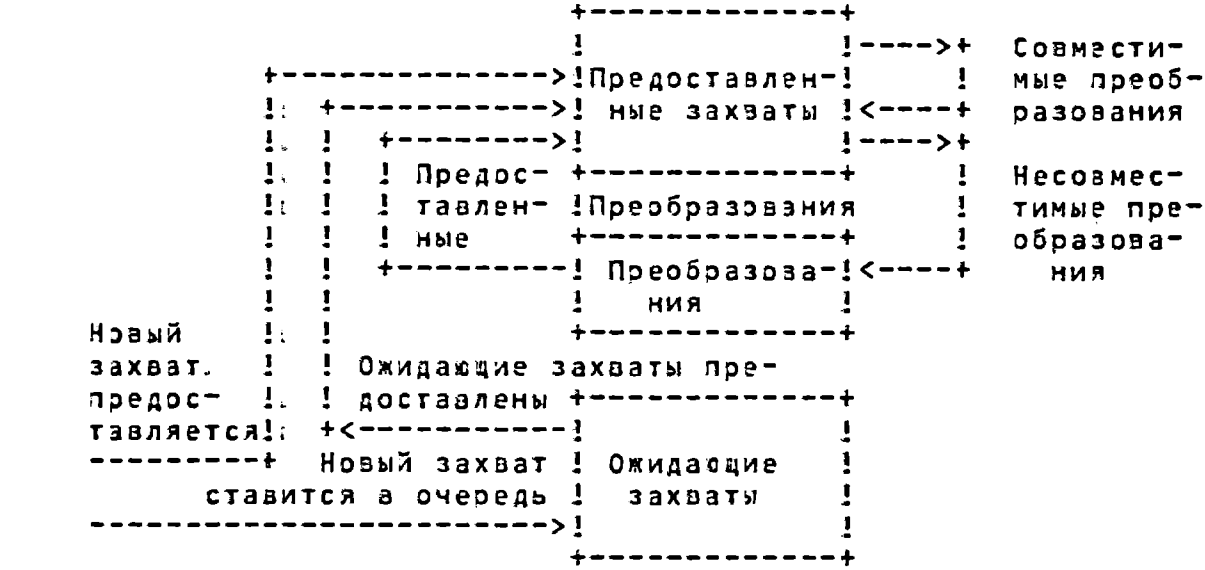


Рис. 46

Когда делается запрос на новый захват, служебные программы управления захватом сначала определяют, известен ли

данный ресурс в настоящий момент (т.е. имеют ли какие-нибудь другие процессы запрос на захват этого ресурса). Если ресурс новый (т.е. не существует других запросов на захват данного ресурса), то служебные программы управления захватом создают элемент для нового ресурса и запрошенного захвата. Если ресурс уже известен, то программы захвата определяют, есть ли какие-нибудь другие запросы на захват либо в очереди ожидающих запросов, либо в очереди преобразований. Если и та, и другая очереди пусты, то служебные программы управления захватом определяют, совместим ли новый захват с другими предоставленными захватами. Если захват совместим, то он предоставляется. Если запрос на захват несовместим, то он помещается в очередь ожидающих запросов. При помощи бита флага можно указать служебным программам управления захватом, что, если запрос нельзя удовлетворить немедленно, то его не требуется помещать в очередь.

12.1.6. Концепции преобразования захвата

Преобразования захвата позволяют процессам изменить уровень захватов. Например, процесс может установить захват низкого уровня на ресурс до тех пор, пока он не захочет ограничить доступ к этому ресурсу. Тогда процесс может запросить преобразование захвата.

Преобразования захвата задают при помощи бита флага (т. 12.3.4) и блока состояния захвата. Блок состояния захвата должен содержать идентификацию преобразуемого захвата.

если новый режим захвата совместим с текущими предоставленными захватами, то преобразованный запрос удовлетворяется сразу же. Если новый режим захвата несовместим с существующими захватами в очереди предоставлечных захватов, то данный запрос помещается в очередь преобразований. Данный захват сохраняет свой старый режим захвата и не получает новый режим захвата до тех пор, пока не будет удовлетворен запрос на преобразование.

Когда захват удаляется из очереди или преобразуется в режим захвата более высокого уровня, служебные программы управления захватом проверяют первый запрос на преобразование в очереди запросов на преобразование. Если запрос на преобразование совместим с предоставленными в настоящий момент захватами, то он удовлетворяется. Также удовлетворяются все непосредственно следующие за ним совместимые запросы на преобразование. Если очередь на преобразование пуста, то проверяется очередь ожидающих запросов. Если первый запрос на захват в очереди ожидающих запросов совместим с предоставленными на текущий момент захватами, то он удовлетворяется. Также удовлетворяются все непосредственно следующие за ним совместимые запросы на захват.

12.1.7. Обнаружение "мертвого захвата"

"мертвый захват" возникает в том случае, когда существует некоторая группа захватов, ожидающих друг друга. Пример такой ситуации показан на рис. 47.

"мертвый захват"

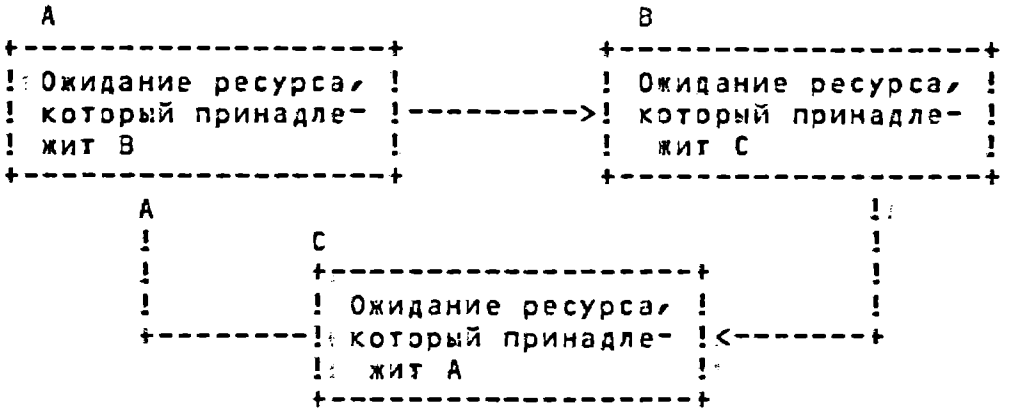


Рис. 47

На рис. 47 три процесса поставили в очередь запросы на ресурсы, к которым нельзя получить доступ до тех пор, пока из очереди не будут удалены текущие задерживающие захваты (или пока они не будут преобразованы в режим захвата более низкого уровня).

Если служебные программы управления захватом определяют, что существует "мертвый захват", то для этого захвата выбирается (программами) некоторый процесс. Если выбранный процесс запрашивал новый захват, то этот захват не предоставляется, если он запрашивал преобразование захвата, то данному захвату возвращается его старый режим захвата. В любом случае в блок состояния захвата помещается код состояния SS$_READLOCK. Предоставленные захваты никогда не отменяются. Только ожидающие запросы на захват могут получить код состояния SS$_READLOCK.

Примечание. Программы пользователя не должны брать на себя решение того, какой процесс будет выбран для разрыва "мертвого захвата".

12.2. Постановка в очередь простых запросов на захват

Для постановки в очередь запросов на захват используется программа системного обслуживания $ENQ. Когда запрашиваются новые захваты, в вызове программы системного обслуживания должны быть указаны режим захвата, адрес блока состояния захвата и имя ресурса. Следующий пример иллюстрирует вызов $ENQ.

Пример.

```
LKSB:     .BLKQ     0              место блока состояния
RESOURCE:                          захвата
          .ASCID /STRUCTURE_1/     STRUCTURE_1 это имя зах-
                                   ватываемого ресурса


          $ENQW_S LKMODE=#LCK$K_PRMODE, -     режим
                                              защищенного чтения
                  LKSB=LKSB,-
                  RESNAM=RESOURCE
```

В данном примере структура данных STRUCTURE_1 доступна нескольким процессам. Некоторые процессы читают эту структуру данных, другие пишут в нее. Структура должна быть защищена от чтения читающими программами на то время, пока она обновляется записывающими программами. В примере читающая программа ставит в очередь запрос на режим защищенного чтения. Режим защищенного чтения совместим с самим собой, поэтому все читающие программы могут читать данную структу-

ру одновременно. Программы, записывающие в эту структуру, используют захваты с режимом защищенной записи или с монопольным режимом. Поскольку режим защищенной записи и монопольный режим несовместимы с режимом защищенного чтения, никакие записывающие программы не могут писать в структуру данных до тех пор, пока читающие программы не отменят свои захваты и никакие читающие программы не могут читать структуру данных пока записывающие программы не отменят свои захваты.

12.3. Синхронизация захватов

Программа системного обслуживания $ENQ возвращает управление вызывающей программе после постановки в очередь запроса на захват. Код состояния в регистре R0 показывает, насколько успешно выполнена постановка запроса в очередь. После постановки запроса в очередь программа не имеет доступа к данному ресурсу до тех пор, пока запрос не будет удовлетворен. Для того, чтобы узнать, удовлетворен ли уже запрос, программа может использовать три способа:

1) задать номер флага события, который должен устанавливаться, когда запрос будет удовлетворен, и ждать установки этого флага события;

2) указать адрес подпрограммы обработки асинхронного системного прерывания (AST), которую необходимо выполнить, когда запрос будет удовлетворен;

3) опрашивать блок состояния захвата о наличии возвращаемого кода состояния, который указывает на то, что запрос

уже удовлетворен.

Эти методы синхронизации идентичны тем способам синхронизации, которые используются программами системного обслуживания $QIO (подраздел 7.3).

Программа $ENQW осуществляет синхронизацию, комбинируя функции программ системного обслуживания $ENQ и $SYNCH. Программа $ENQW имеет те же самые аргументы, что и программа $ENQ. Она ставит в очередь запрос на захват, а затем переводит программу в состояние ожидания флага события (LEF) до тех пор, пока не будет удовлетворен запрос на захват.

12.3.1. Извещение о синхронизации выполнения

Служебные программы управления захватом обеспечивают механизм, который позволяет процессам определить, что запрос на захват удовлетворен синхронно, то есть запрос на захват не помещался в очередь ожидающих запросов или в очередь запросов на преобразование.

Если установлен бит флага LCK$M_SYNCSTS, и захват предоставлен синхронно, то в регистр R0 возвращается код состояния SS$_SYNCH, не устанавливаются никакие флаги события и не осуществляется доставка AST.

Если запрос не выполняется синхронно, то возвращается код успешного завершения SS$_NORMAL. Флаги событий или подпрограммы AST работают как обычно (т.е., когда захват будет предоставлен, устанавливается флаг события и доставляется AST).

13.63. $GETQUI - получить информацию об очереди

Программа системного обслуживания $GETQUI возвращает информацию об очередях и заданиях, инициируемых из данных очередей. Программы системного обслуживания $GETQUI и $SNDJBC вместе обеспечивают пользователя интерфейсом с контроллером заданий операционной системы МОС ВП, который является программой управления очередями и учетной информацией.

Программа $GETQUI завершается асинхронно, то есть она возвращает управление вызывающему процессу после постановки запроса в очередь, не ожидая завершения операции.

Для синхронного завершения следует использовать программу системного обслуживания "получить информацию об очереди и ждать" ($GETQUIW). Программа $GETQUIW полностью идентична программе $GETQUI за исключением того, что программа $GETQUIW возвращает управление вызывающему процессу после завершения операции (см. подраздел 2.5).

Формат:

SYS$GETQUI [EFN],FUNC,[NULLARG],[ITMLST],
[IOSB],[ASTADR],[ASTPRM]

Аргументы:

EFN

Использование в МОС ВП:	EF_NUMBER
Тип:	длинное слово (без знака)
Доступ:	только запись
Механизм:	по значению

Номер флага события, который устанавливается после завершения программы $GETQUI. Аргумент EFN является длинным словом, содержащим данный номер. Аргумент EFN является необязательным аргументом.

Когда запрос ставится в очередь, программа $GETQUI сбрасывает указанный флаг события (или флаг события 0, если не задан аргумент EFN). Затем после завершения операции программа $GETQUI устанавливает указанный флаг события (или флаг события 0).

FUNC

Использование в МОС ВП: FUNCTION_CODE

Тип: слово (без знака)

Доступ: только чтение

Механизм: по значению

Код функции, указывающий, какую функцию должна выполнять программа $GETQUI. Аргумент FUNC является словом, содержащим код функции.

Для одного вызова программы $GETQUI можно задать только один код функции. Большинство кодов функций требуют и допускают задание дополнительной информации, которая передается при вызове программы. Данная информация передается с помощью аргумента ITMLST, который задает список из одного или более дескрипторов элементов. Каждый дескриптор элемента, в свою очередь задает код элемента, который либо описывает конкретную информацию, которую должна возвращать программа $GETQUI, либо как-то влияет на действия, задаваемые кодом функции.

В следующем списке перечислены все коды функций, описаны действия, которые они задают, и указаны относящиеся к ним коды элементов. Описания данных кодов элементов приведено в пункте, касающемся аргумента ITMLST.

Коды функций программы ¤GETQUI и допустимые коды элементов:

QUI¤_CANCEL_OPERATION -

Данный запрос прекращает любую операцию в режиме поиска по случайной последовательности запросов, инициированную предыдущим вызовом программы ¤GETQUI, освобождая при этом блок контекста GETQUI (GQC), который система поддерживает для процесса пользователя. Поскольку в любой момент времени может быть только одна необработанная случайная последовательность поиска, то нет необходимости задавать какие-либо коды элементов.

Когда пользователь вызывает программу ¤GETQUI, чтобы выполнить серию неупорядоченных запросов для получения информации о характеристиках, формах или очередях (и связанных с ними заданиях и файлах), контроллер заданий между вызовами формирует блоки, которые указывают на следующий об'ект в случайной последовательности запросов. Система сохраняет данную информацию до тех пор, пока:

1) пользователь вызвал программу ¤GETQUI, чтобы проверить каждый об'ект в последовательности;

2) процесс пользователя был прекращен;

3) пользователь явно прекратил операцию последовательностью запросов, используя функциональный код

QUI¤_CANCEL_OPERATION.

Если вызовы программы ¤GETQUI обнаружили все об'екты в той последовательности, в которой они интересуют пользователя, то следует прекратить операцию со случайной последовательностью запросов. Это освобождает ресурсы контроллера заданий и позволяет пользователю инициировать другую операцию ¤GETQUI;

QUI¤_DISPLAY_CHARACTERISTIC -

Данный запрос возвращает информацию об определении конкретной характеристики или определении следующей характеристики в операции последовательностью запросов. Успешная операция последовательностью запросов для данного кода функции завершается, когда программа ¤GETQUI вернет информацию обо всех определениях характеристик, включенных в последовательность. Программа системного обслуживания ¤GETQUI сигнализирует о завершении данной последовательчости запросов, возвращая значение кода состояния JBC¤_NOMORECHAR в блоке состояния ввода-вывода. Если программа системного обслуживания ¤GETQUI не находит ни одного определения характеристики, то она возвращает в блоке состояния ввода-вывода значение кода состояния JBC¤_NOSUCHCHAR.

Необходимо задать один из следующих входных кодов элемента. (Можно задать оба кода):

1) QUI¤_SEARCH_NAME;

2) QUI¤_SEARCH_NUMBER.

Можно задать следующий входной код элемента:

QUI¤_SEARCH_FLAGS.

Можно задать следующие выходные кода элемента:

1) QUI¤_CHARACTERISTIC_NAME;

2) QUI¤_CHARACTERISTIC_NUMBER;

QUI¤_DISPLAY_FILE -

Данный запрос обычно делается как часть операций последовательностью запросов об очередях, заданиях, файлах. Это означает, что пользователь перед тем, как сделать вызов программы ¤GETQUI, чтобы запросить информацию о файле, уже сделал два предварительных вызова данной программы, чтобы установить очередь, контекст заданий и задание, которые содержат файлы, интересующие пользователя. В данном случае программа системного обслуживания ¤GETQUI возвращает информацию о следующем файле, определенном для текущего контекста задания. Программа ¤GETQUI сигнализирует, что она вернула информацию обо всех файлах, определенных для текущего контекста задания, возвращая значение кода состояния JBC¤_NOMOREFILE в блоке состояния ввода-вывода.

Если текущий контекст задания не содержит файлов, то программа возвращает в блоке состояния ввода-вывода значе-

ние кода состояния JBC¤_NOSUCHFILE. Пакетное задание может вызвать программу ¤GETQUI, чтобы запросить информацию о командном файле, который выполняется в текущий момент, не делая предварительно вызовов данной программы системного обслуживания для установления очереди и контекста задания. Чтобы это выполнить, пакетное задание определяет возможность QUI¤V_SEARCH_THIS_JOB кода элемента QUI¤_SEARCH_FLAGS. Операционная система МОС ЭП не сохраняет очереди или контекста задания, установленных в таком вызове.

Можно задать следующий входной код элемента.

QUI¤_SEARCH_FLAGS.

Можно задать следующие выходные коды:

1) QUI¤_FILE_COPIES;

2) QUI¤_FILE_COPIES_DONE;

3) QUI¤_FILE_FLAGS;

4) QUI¤_FILE_SETUP_MODULES;

5) QUI¤_FILE_SPECIFICATION;

6) QUI¤_FILE_STATUS;

7) QUI¤_FIRST_PAGE;

8) QUI¤_LAST_PAGE;

QUI¤_DISPLAY_FORM -

Данный запрос возвращает информацию о конкретном определении формы или следующем определении формы в операции последовательностью запросов. Успешная операция с последовательностью запросов для данного кода функции завершается, когда программа ¤GETQUI возвращает инфор-

мацию обо всех определениях формы включенных в последовательность запросов. Программа системного обслуживания ¤GETQUI сигнализирует о завершении данной последовательности запросов, возвращая в блок состояния ввода-вывода значение кода завершения JBC¤_NOMOREFORM. Если программа не находит ни одного определения формы, то она возвращает в блоке состояния ввода-вывода значение кода состояния JBC¤_NOSUCHFORM.

Необходимо задать один из следующих входных элементов кода. (Можно задать оба данных кода):

1) QUI¤_SEARCH_NAME;

2) QUI¤_SEARCH_NUMBER.

Можно задать следующий входной код элемента:

QUI¤_SEARCH_FLAGS.

Можно задать следующие выходные коды элементов:

1) QUI¤_FORM_DESKRIPTION;

2) QUI¤_FORM_FLAGS;

3) QUI¤_FORM_LENGTH;

4) QUI¤_FORM_MARGIN_BOTTOM;

5) QUI¤_FORM_MARGIN_LEFT;

6) QUI¤_FORM_MARGIN_RIGTH;

7) QUI¤_FORM_MARGIN_TOP;

8) QUI¤_FORM_NAME;

9) QUI¤_FORM_NUMBER;

10) QUI¤_FORM_SETUP_MODULES;

11) QUI¤_FORM_STOCK;

12) QUI¤_FORM_WIDTH;

13) QUI¤_PAGE_SETUP_MODULES;

QUI¤_DISPLAY_JOB -

Данный запрос обычно делается как часть последовательности операций с очередями и заданиями, то есть перед тем как сделать вызов программы ¤GETQUI, чтобы запросить информацию о задании, делается вызов данной служебной программы для установления контекста очереди, содержащей задание, интересующее пользователя. В данном случае программа возвращает информацию о следующем задании, определенном для текущего контекста очереди. Операция QUI¤_DISPLAY_FILE также устанавливает контекст задания для последующей операции QUI¤_DISPLAY_JOB. Данный контекст задания сохраняется до тех пор, пока не будет сделан другой вызов программы ¤GETQUI с кодами функций QUI¤_DISPLAY_JOB, QUI¤_DISPLAY_QUEUE или QUI¤_CANCEL_OPERATION. Программа сигнализирует, что она вернула информацию обо всех заданиях, содержащихся в текущем контексте очереди, возвращая в блоке состояния ввода-вывода значение кода состояния JBC¤_NOMOREFILE. Если текущий контекст очереди не содержит заданий, то программа возвращает в блок состояния ввода-вывода значение кода состояния JBC¤_NOSUCHFILE. Пакетное задание может сделать вызов программы системного обслуживания ¤GETQUI, чтобы запросить информацию о самом себе, не делая предварительно вызова данной программы для установления контекста очереди. Чтобы выполнить это, пакетное задание должно

задать возможность QUI¤V_SEARCH_THIS_JOB кода элемента QUI¤_SEARCH_FLAGS. Операционная система МОС ВП не сохраняет контекста задания или очереди, установленного в таком вызове.

Можно задать следующий входной код элемента:

QUI¤_SEARCH_FLAGS.

Можно указать следующие входные коды элементов:

1) QUI¤_ACCOUNT_NAME;

2) QUI¤_AFTER_TIME;

3) QUI¤_CHARACTERISTICS;

4) QUI¤_CHECKPOINT_DATA;

5) QUI¤_CLI;

6) QUI¤_COMPLETED_BLOCKS;

7) QUI¤_CONDITION_VECTOR;

8) QUI¤_CPU_LIMIT;

9) QUI¤_ENTRY_NUMBER;

10) QUI¤_FORM_NAME;

11) QUI¤_INTERVENING_BLOCKS;

12) QUI¤_INTERVENING_JOBS;

13) QUI¤_JOB_COPIES_DONE;

14) QUI¤_JOB_FLAGS;

15) QUI¤_JOB_NAME;

16) QUI¤_JOB_SIZE;

17) QUI¤_JOB_STATUS;

18) QUI¤_LOG_QUEUE;

19) QUI¤_LOG_SPECIFICATION;

20) QUI¤_NOTE;

21) QUI$_OPERATOR_REQUEST;

22) QUI$_PARAMETER_1 до 8;

23) QUI$_PRIORITY;

24) QUI$_QUEUE_NAME;

25) QUI$_REQUEUE_QUEUE_NAME;

25) QUI$_SUBMISSION_TIME;

27) QUI$_UIC;

28) QUI$_USERNAME;

29) QUI$_WSDEFAULT;

30) QUI$_WSEXTENT;

31) QUI$_WSQUOTA;

QUI$_DISPLAY_QUEUE -

Данный запрос возвращает информацию о конкретном определении очереди или следующем определении очереди в операции с последовательностью запросов. Если вызов успешен, то данная операция также устанавливает контекст очереди для последующей операции QUI$_DISPLAY_JOB. Установленный контекст очереди сохраняется до тех пор, пока не будет сделан другой вызов программы $GETQUI с кодами функций QUI$_DISPLAY_QUEUE или QUI$_CANCEL_OPERATION. Программа $GETQUI сигнализирует, что она вернула информацию обо всех очередях, содержащихся в текущей последовательности запросов, возвращая в блоке состояния ввода-вывода значение кода состояния JBC$_NOMOREQUE. Если не найдено ни одной очереди, то программа возвращает в блоке состояния ввода-вывода значение кода состояния JBC$_NOSUCHQUE.

Пакетное задание может сделать вызов программы ¤GETQUI для запроса информации об очереди, в которой оно содержится, не делая предварительного вызова данной программы для установления контекста очереди. Для этого пакетное задание должно указать возможность QUI¤V_SEARCH_THIS_JOB кода элемента QUI¤_SEARCH_FLAGS. Операционная система МОС ВП не сохраняет контекста очереди, установленного для такого вызова.

Необходимо задать следующий входной код элемента:

QUI¤_SEARCH_NAME.

Можно задать следующий входной код элемента:

QUI¤_SEARCH_FLAGS.

Можно задать следующие выходные коды элементов:

1) QUI¤_ASSIGNED_QUEUE_NAME;

2) QUI¤_BASE_PRIORITY;

3) QUI¤_CHARACTERISTICS;

4) QUI¤_CPU_DEFAULT;

5) QUI¤_CPU_LIMIT;

6) QUI¤_DEVICE_NAME;

7) QUI¤_FORM_NAME;

8) QUI¤_GENERIC_TARGET;

9) QUI¤_JOB_LIMIT;

10) QUI¤_JOB_RESET_MODULES;

11) QUI¤_JOB_SIZE_MAXIMUM;

12) QUI¤_JOB_SIZE_MINIMUM;

13) QUI¤_LIBRARY_SPECIFICATION;

14) QUI¤_OWNER_UIC;

15) QUI¤_PROCESSOR;

16) QUI¤_PROTECTION;

17) QUI¤_QUEUE_FLAGS;

18) QUI¤_QUEUE_NAME;

19) QUI¤_QUEUE_STATUS;

20) QUI¤_SCSNODE_NAME;

21) QUI¤_WSDEFAULT;

22) QUI¤_WSEXTENT;

23) QUI¤_WSQUOTA;

QUI¤_TRANSLATE_QUEUE -

Данный запрос транслирует логическое имя для очереди в эквивалентное имя для данной очереди. Логическое имя задается кодом QUI¤_SEARCH_NAME. Трансляция выполняется итеративно, пока либо не будет найдена эквивалентная строка, либо не будет достигнуто число трансляций, допустимое системой.

Необходимо задать следующий входной код элемента:

QUI¤_SEARCH_NAME.

Можно указать следующий выходной код элемента:

QUI¤_QUEUE_NAME.

NULLARGE

Использование в МОС ВП: NULL_ARG

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Аргумент, занимающий место.

ITMLST

Использование в МОС ВП: ITEM_LIST_3

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Список элементов, содержащий информацию, которая должна использоваться при выполнении функции, заданной аргументом FUNC. Аргумент ITMLST является адресом данного списка элементов. Список элементов состоит из одного или более дескрипторов элементов, каждый из которых задает код элемента. Список элементов завершается кодом элемента 0 или длинным словом содержащим 0. Структура дескриптора элемента описана на рис.67.

Структура дескриптора элемента

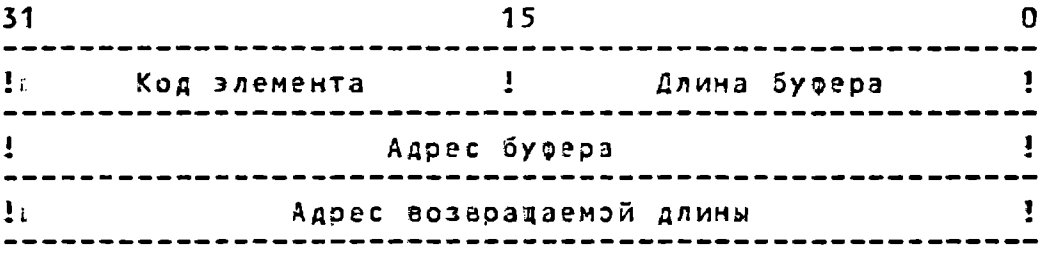


Рис. 67

Поля дескрипторов элементов программы ¤GETQUI:

1) длина буфера - слово, задающее длину буфера.

Данный буфер либо предоставляет информацию, которая должна использоваться программой ¤GETQUI, либо получает информацию от данной программы. Требуемая длина буфера меняется в зависимости от заданного кода элемента и указана

в описании каждого кода элемента;

2) код элемента - слово, содержащее код элемента, который определяет сущность информации, используемой программой ¤GETQUI, либо получаемой от данной программы. Каждый код элемента имеет символическое имя. Данные символические имена определяются макрокомандой ¤QUIDEF и имеет форму QUI¤_код.

Имеется два типа кодов элементов:

1) код входного значения.

Программа ¤GETQUI имеет всего только три кода элементов входного значения: QUI¤_SEARCH_NAME, QUI¤_SEARCH_NUMBER и QUI¤_SEARCH_FLAGS. Данные коды элементов задают имя или номер об'екта, для которого программа должна вернуть информацию, и указывают пределы, в которых программа ¤GETQUI должна искать данные об'екты. Большинство кодов функций требуют задания по крайней мере одного кода элемента входного значения. В описании каждого кода функции перечислены обязательные и необязательные коды элементов входного значения.

Для кодов элементов входного значения поля "длина буфера" и "адрес буфера" дескриптора элемента должны быть ненулевыми. Поле "адрес возвращаемой длины" должно быть нулевым. Конкретные требования к длине буфера приведены в описании каждого кода элемента;

2) код выходного значения.

Коды элементов выходного значения задают буфер, в который программа ¤GETQUI возвращает информацию. Для данных кодов поля "длина буфера" и "адрес буфера" дескриптора элемента должны быть ненулевыми. Поле "адрес возвращаемой длины" может быть нулевым или ненулевым. Конкретные требования к длине буфера приведены в описании каждого элементного кода.

Некоторые коды элементов задают имя очереди, имя формы или имя характеристики для программы ¤GETQUI или требуют, чтобы программа вернула одно из данных имен. Для таких кодов элементов буфер должен задавать или быть готовым принять строку, содержащую от 1 до 31 символа, исключая пробелы, знаки табуляции или неопределенные символы, которые игнорируются. Допустимыми символами в строке являются прописные буквы, строчные буквы (которые преобразуются в прописные), цифры, знак денежной единицы (¤) и знак подчеркивания (_);

3) адрес буфера.

Который задает или получает информацию;

4) адрес возвращаемой длины.

Адрес слова, в которое помещается длина информации, возвращаемой программой ¤GETQUI.

Коды элементов программы ¤GETQUI:

QUI¤_ACCOUNT_NAME -

Если задан данный код элемента, то программа возвращает 8-байтовое учетное имя владельца указанного зада-

ния. (QUI¤_DISPLAY_JOB код элемента);

QUI¤_AFTER_TIME -

Если задан данный код элемента, то программа возвращает системное время в формате квадрослова абсолютного значения времени, в которое или после которого может выполняться указанное задание. (QUI¤_DISPLAY_JOB код элемента);

QJI¤_ASSIGNED_QUEUE_NAME -

Если задан данный код элемента (для функции QUI¤_DISPLAY_QUEUE), то программа возвращает в виде строки символов имя исполняемой очереди, которой назначена логическая очередь, заданная в вызове программы ¤GETQUI. Поскольку имя очереди может включать до 31 символа, то поле "длина буфера" дескриптора элемента должно задавать 31 байт;

QUI¤_BASE_PRIORITY -

Если задан данный код элемента (для функции QUI¤_DISPLAY_QUEUE), то программа возвращает в длинном слове значение от 0 до 15, означающее приоритет, с которым пакетные задания инициируются из исполняемой очереди пакета, или приоритета симбионтного процесса, который управляет входными исполняемыми очередями;

QJI¤_CHARACTERISTIC_NAME -

Если задан данный код элемента (для функции QUI¤_DISPLAY_CHARACTERISTIC), то прогоамма возвращает в длинном слове значение в диапазоне от 0 до 127, номер указанной характеристики;

QUI¤_CHARACTERISTICS -

Если задан данный код элемента (для функций QUI¤_DISPLAY_JOB, QUI¤_DISPLAY_QUEUE), то программа возвращает в виде 128-битовой строки (16-байтовое поле) характеристики, связанные с указанной очередью или заданием. Каждый установленный бит в маске битов представляет номер характеристики в диапазоне от 0 до 127;

QUI¤_CHECKPOINT_DATA -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает в виде строки символов значение символического имени BATCH¤RESTART языка DCL при повторном старте указанного пакетного задания. Поскольку значение данного символического имени может включать в себя до 255 символов, то в поле "длина буфера" дескриптора элемента следует задавать 255 байтов;

QUI¤_CLI -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает имя файла интерпретатора команд языка команд, используемого для выполнения указанного пакетного задания. Возвращаемая спецификация файла предполагает имя устройства SYS¤SYSTEM и тип файла EXE. Поскольку имя файла может включать в себя до 39 символов, то в поле "длина буфера" дескриптора элемента следует задавать 39 байтов. Данный код относится только к пакетным заданиям;

QUI¤_COMPLETED_BLOCKS -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает в длинном слове десятичное значение, которое представляет число блоков, которые обработаны симбионтом для указанного задания на печать. Данный код элемента относится только к заданиям на печать;

QUI¤_CONDITION_VECTOR -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает в длинном слове значение кода завершения указанного задания;

QUI¤_CPUDEFAULT -

Если задан данный код элемента (для функции QUI¤_DISPLAY_QUEUE), то программа возвращает в длинном слове десятичное значение, которое представляет принятый по умолчанию лимит времени центрального процессора, заданный для очереди в 10 миллисекундных интервалах. Данный код относится только в пакетным исполняемым очередям;

QUI¤_CPU_LIMIT -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB, DISPLAY_QUEUE), то программа возвращает в длинном слове десятичное значение, которое представляет максимальный лимит времени центрального процессора, заданный для указанного задания или очереди в 10-миллисекундных интервалах. Данный код относится только к пакетным заданиям и к пакетным исполняемым

очередям;

QJI¤_DEVICE_NAME -

Если задан данный код элемента (для функции QUI¤_DISPLAY_QUEUE), то программа возвращает строку символов длиной от 1 до 31, представляющее устройство, на котором локализована указанная исполняемая очередь;

QUI¤_ENTRY_NUMBER -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает в длинном слове десятичное значение, которое представляет номер элемента очереди для указанного задания;

QJI¤_FILE_COPIES -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FILE), то программа возвращает в длинном слове десятичное значение от 1 до 255, указывающее сколько раз необходимо обрабатывать данный файл. Данный код относится только к выходным исполняемым очередям;

QJI¤_FILE_FLAGS -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FILE), то программа возвращает в длинном слове вектор битов, представляющих варианты обработки, выбранные для указанного файла. Каждому варианту обработки соответствует один бит. Если программа ¤GETQUI устанавливает бит, то файл обрабатывается согласно соответствующему варианту обработки. Каждый бит в векторе имеет символическое имя.

Символические имена (определяются макрокомандой ¤QUIDEF):

1) QUI¤V_FILE_BURST - страницы пакетов и флагов должны быть напечатаны перед файлом;

2) QUI¤V_FILE_DELETE - файл должен быть удален после выполнения запроса;

3) QUI¤V_FILE_DOUBLE_SPACE - симбионт форматирует файл с удвоением пробелов;

4) QUI¤V_FLAG - страница флагов должна быть напечатана перед файлом;

5) QUI¤V_FILE_TRAILER - дополнительная страница должна быть напечатана после файла;

6) QUI¤V_FILE_PAGE_HEADER - заголовок страницы должен печататься на каждой выводимой странице;

7) QUI¤V_FILE_PAGINATE - симбионт нумерует страницы, выводя подачу формы, как только вывод достигает нижней границы формы;

8) QUI¤V_FILE_PASSALL - симбионт печатает файл в в режиме PASSALL;

QUI¤_FILE_SETUP_MODULES -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FILE), то программа возвращает список имен (через запятую) текстовых модулей, которые должны быть извлечены из библиотеки управления устройствами и распечатаны на печатающем устройстве перед печатью указанного файла. Поскольку имя текстового модуля может включать в себя до 31 символа и отделяться от

предыдущего имени текстового модуля запятой, то поле "длина буфера" дескриптора элемента должно задавать 32 байта для каждого возможного текстового модуля. Данный код имеет смысл только для выходных исполняемых очередей;

QUI¤_FILE_SPECIFICATION -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FILE), то программа возвращает полностью квалифицированную спецификацию файла, информацию о котором запрашивает пользователь. Поскольку спецификация файла может включать в себя до 255 символов, то поле "длина буфера" дескриптора элемента должно задавать 255 байтов;

QUI¤_FILE_STATUS -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FILE), то программа возвращает информацию о состоянии файла в виде векторов битов размером в длинное слово. Каждый код состояния файла представлен одним битом. Каждый бит в векторе имеет свое символическое имя.

Символические имена (определяются макрокомандой ¤QUIDEF):

1) QUI¤V_FILE_CHECKPOINT - файл копируется в контрольных точках;

2) QUI¤V_FILE_EXECUTING - файл обрабатывается;

QUI¤_FIRST_PAGE -

Если задан данный код элемента (для функции

QUI¤_DISPLAY_FILE), то программа возвращает в длинном слове десятичное значение, представляющее номер страницы, с которой должна начинаться печать указанного файла. Данный код относится только к выходным исполняемым очередям;

QUI¤_FORM_DESCRIPTION -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FORM), то программа возвращает в виде строки символов текст, описывающий для пользователей и операторов указанную форму. Поскольку данная текстовая строка может включать в себя до 255 символов, то поле "длина буфера" дескриптора элемента должно задавать 255 байтов;

QUI¤_FORM_FLAGS -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FORM), то программа возвращает в длинном слове вектор битов, представляющих варианты обработки, выбранные для указанной формы. Каждому варианту обработки соответствует один бит. Если программа ¤GETQUI устанавливает какой-то бит, то форма обрабатывается согласно соответствующему варианту обработки. Каждый бит в векторе имеет свое символическое имя.

Символические имена (определяются макрокомандой ¤QUIDEF):

1) QUI¤V_FORM_SHEET_FEED - симбионт делает паузу в конце каждой физической страницы, так что можно вставить новый лист бумаги;

2) QUI$V_FORM_TRUNCATE - печатающее устройство отвергает все символы, которые выходят за пределы заданной правой границы;

3) QUI$V_FORM_WRAP - печатающее устройство печатает на следующей строке символы, выходящие за пределы заданной правой границы;

QUI$_FORM_LENGTH -

Если задан данный код элемента (для функции QUI$_DISPLAY_FORM), то программа возвращает в длинном слове десятичное значение, представляющее физическую длину указанной формы в строках. Данный код относится только к выходным очередям;

QUI$_FORM_MARGIN_BOTTOM -

Если задан данный код элемента (для функции QUI$_DISPLAY_FORM),то программа возвращает в длинном слове десятичное значение, представляющее нижнюю границу указанной формы в строках;

QUI$_FORM_MARGIN_LEFT(RIGHT) -

Если задан данный код элемента (для функции QUI$_DISPLAY_FORM), то программа возвращает в длинном слове десятичное значение, представляющее левую (правую) границу указанной формы в символах;

QUI$_FORM_MARGIN_TOP -

Если задан данный код элемента (для функции QUI$_DISPLAY_FORM), то программа возвращает в длинном слове десятичное значение, представляющее верхнюю границу указанной формы в строках;

QUI¤_FORM_NAME -

Если задан данный код элемента (для функций QUI¤_DISPLAY_FORM, QUI¤_DISPLAY_JOB, QUI¤_DISPLAY_QUEUE), то программа возвращает в виде строки символов имя указанной формы или формы, связанной с указанным заданием или очередью. Поскольку имя формы может включать в себя до 31 символа, то поле "длина буфера" дескриптора элемента должно задавать 31 байт;

QUI¤_FORM_NUMBER -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FORM), то программа возвращает в длинном слове десятичное значение, представляющее номер указанной формы;

QUI¤_FORM_SETUP_MODULES -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FORM), то программа возвращает список имен (разделенных запятыми) текстовых модулей, которые должны быть извлечены из библиотеки управления устройствами и распечатаны на печатающем устройстве перед печатью файла, на указанной форме. Поскольку имя текстового модуля может включать в себя до 31 символа плюс запятая, отделяющая его от предыдущего имени текстового модуля, то поле "длина буфера" дескриптора элемента должно задавать 32 байта для каждого возможного текстового модуля. Данный код имеет смысл только для выходных исполняемых очередей;

QUI¤_FORM_STOCK -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FORM), то программа возвращает в виде строки символов имя типа бумаги, на которой должна печататься указанная форма.поскольку имя типа бумаги может включать в себя до 31 символа, то поле "длина буфера" дескриптора элемента должно задавать 31 байт;

QUI¤_FORM_WIDTH -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FORM), то программа возвращает в длинном слове десятичное значение, представляющее ширину указанной формы в символах;

QUI¤_GENERIC_TARGET -

Если задан данный код элемента (для функции QUI¤_DISPLAY_QUEUE), то программа возвращает список имен (через запятую) исполняемых очередей, которым разрешено принимать работу из указанной общей очереди. Поскольку имя очереди может включать в себя до 31 символа плюс запятая, отделяющая его от предыдущего имени очереди, то поле "длины буфера" дескриптора элемента должно задавать 32 байта. Общая очередь может посылать работу максимум 124 исполняемым очередям. Данный код имеет смысл только для общих очередей;

QUI¤_INTERVENING_BLOCKS -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает в длинном слове десятичное значение, представляющее число бло-

ков, которые должны быть обработаны перед началом выполнения указанного задания. данный код имеет смысл только для выходных исполняемых очередей;

QUI¤_INTERVENING_JOBS -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает в длинном слове десятичное значение, представляющее число заданий, которые должны быть обработаны перед началом выполнения указанного задания. Данный код имеет смысл только для выходных исполняемых очередей;

QUI¤_JOB_COPIES -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает в длинном слове десятичное число, означающее сколько раз должно быть повторено указанное задание на печать;

QUI¤_JOB_FLAGS -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает в длинном слове вектор битов, представляющих варианты обработки, выбранные для указанного задания. Каждый вариант обработки представлен одним битом. Если программа ¤GETQUI устанавливает какой-то бит, то задание обрабатывается согласно соответствующему варианту обработки. Каждый бит в векторе имеет свое символическое имя.

Символические имена (определяются макрокомандой $QUIDEF):

1) QUI$V_JOB_CPU_LIMIT - для данного задания задан лимит времени центрального процессора;

2) QUI$V_JOB_FILE_BURST - каждому файлу в задании предшествуют страницы пакета и флагов;

3) QUI$V_JOB_FILE_BURST_ONE - страницы пакета и флагов предшествуют только первой копии первого файла в задании;

4) QUI$V_JOB_FILE_FLAG - страница флагов предшествует каждому файлу в задании;

5) QUI$V_JOB_FILE_FLAG_ONE - страница флагов предшествует только первой копии первого файла в задании;

6) QUI$V_JOB_FILE_PAGINATE - симбионт нумерует выходные страницы, вводя подачу формы, как только вывод достигнет нижней границы формы;

7) QUI$V_JOB_FILE_TRAILER - дополнительная страница следует за каждым файлом в задании;

8) QUI$V_JOB_FILE_TRAILER_ONE - дополнительная страница следует только за последней копией последнего файла в задании.

9) QUI$V_JOB_LOG_DELETe - регистрационный журнал удаляется после того, как он будет напечатан.

10) QUI$V_JOB_LOG_NULL - регистрационный файл не создан;

11) QUI$V_JOB_LOG_SPOOL - регистрационный файл задания поставлен в очередь для печати после завершения задания;

12) QUI$V_JOB_LOWERCASE - задание должно печататься на

устройстве печати, которое может печатать как прописные так и строчные буквы;

13) QUI$V_JOB_NOTIFY - когда выполнение задания завершается или отменяется, на терминал отправляется сообщение;

14) QUI$V_JOB_RESTART - после сбоя системы будет выполнен повторный старт задания или может быть заново поставлено в очередь во время выполнения;

15) QUI$V_JOB_WSDEFAULT - для задания указан размер рабочего набора, принятый по умолчанию.

16) QUI$V_JOB_WSEXTENT - для задания указан экстент рабочего набора;

17) QUI$V_JOB_WSQUOTA - для задания указана квота рабочего набора;

QUI$_JOB_LIMIT -

Если задан данный код элемента (для функции QUI$_DISPLAY_QUEUE), то программа возвращает число заданий, которые могут выполнятся одновременно в указанной очереди. Данное число возвращается в длинном слове в виде десятичного значения в диапазоне от 1 до 255. Данный код относится только к пакетным исполняемым очередям;

QUI$_JOB_NAME -

Если задан данный код элемента (для функции QUI$_DISPLAY_JOB), то программа возвращает в виде строки символов имя указанного задания. Поскольку имя задания может включать в себя до 39 символов, то поле "длина буфера" дескриптора элемента должно задавать 39

байтов;

QJI¤_JOB_RESET_MODULES -

Если задан данный код элемента (для функции QUI¤_DISPLAY_QUEUE), то программа возвращает список имен (через запятую) текстовых модулей, которые должны быть извлечены из библиотеки управления устройствами и напечатаны на устройстве печати перед печатью каждого задания в указанной очереди. Поскольку имя текстового модуля может включать в себя до 31 символ плюс запятая, отделяющая его от предыдущего имени текстового модуля, то поле "длина буфера" дескриптора элемента должно задавать 32 байта для каждого возможного текстового модуля. Данный код имеет смысл только для выходных исполняемых очередей;

QJI¤_JOB_SIZE -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает в длинном слове десятичное значение, представляющее общее число блоков в указанном задании на печать;

QJI¤_JOB_SIZE_MAXIMUM(MINIMUM) -

Если задан данный код элемента (для функции QUI¤_DISPLAY_QUEUE), то программа возвращает в длинном слове десятичное значение, представляющее максимальное (минимальное) число блоков, которое может содержать задание на печать, инициируемое из указанной очереди. Данный код относится только к выходным исполняемым очередям;

QUI¤_JOB_STATUS -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает флаги состояния указанного задания, которые содержаться в векторе битов размером в длинное слово.

Символические имена для данных флагов, которые определяются макрокомандой ¤QUIDEF следующие:

1) QUI¤V_JOB_ABORTING - система пытается отменить выполнение задания;

2) QUI¤V_JOB_EXECUTING - задание выполняется или печатается;

3) QUI¤V_JOB_HOLDING - задание будет задержано до тех пор, пока не будет явно освобождено;

4) QUI¤V_JOB_INACCESSIBLE - вызывающий процесс не имеет доступа с чтением информации указанного задания и файла в системном файле очередей. Следовательно, операции QUI¤_DISPLAY_FILE и QUI¤_DISPLAY_JOB могут возвращать информацию только для следующих кодов элементов выходных значений:

- QUI¤_AFTER_TIME;
- QUI¤_COMPLETED_BLOCKS;
- QUI¤_ENTRY_NUMBER;
- QUI¤_INTERVENING_BLOCKS;
- QUI¤_INTERVENING_JOBS;
- QUI¤_JOB_SIZE;
- QUI¤_JOB_STATUS;

5) QUI¤V_JOB_REFUSED - задание было отвергнуто сим-

бионтом и ждет, пока симбионт примет его для обработки;

6) QUI¤V_JOB_RETAINED - задание завершилось, но оставлено в очереди;

7) QUI¤V_JOB_STARTING - контроллер заданий начал обработку задания и начал обмен информацией с симбионтом вывода;

8) QUI¤V_JOB_TIMED - задание ждет определенного момента времени, чтобы начать выполнение;

QUI¤_LAST_PAGE -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FILE), то программа возвращает в длинном слове десятичное значение, представляющее номер страницы, на которой следует закончить печать указанного файла. Данный код относится только к выходным выполняемым очередям;

QUI¤_LIBRARY_SPECIFICATION -

Если задан данный код элемента (для функции QUI¤_DISPLAY_QUEUE), то программа возвращает имя файла библиотеки управления устройствами для указанной очереди. Спецификация библиотеки предполагает устройство и имя каталога SYS¤LIBRARY и тип файла TLB. Поскольку имя файла может включать в себя до 39 символов, то поле "длина буфера" дескриптора элемента должно задавать 39 байтов. Данный код имеет смысл только для выходных исполняемых очередей;

QUI¤_LOG_QUEUE -

Если задан данный код элемента (для функции

QUI¤_DISPLAY_JOB), то программа возвращает в виде строки символов имя очереди, в которую должен быть введен для печати регистрационный файл, сформированный для указанного пакетного задания. Данный код относится только к пакетным заданиям. Поскольку имя очереди может содержать до 31 символа, то поле "длина буфера" дескриптора элемента должно задавать 31 байт;

QJI¤_LOG_SPECIFICATION -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает имя регистрационного файла, который формируется для данного задания. Поскольку спецификация файла может включать в себя до 255 символов, то поле "длина буфера" дескриптора элемента должно задавать 255 байтов. Данный код имеет смысл только для пакетных заданий. Возвращаемая строка представляет из себя просто спецификацию регистрационного журнала, который был сформирован для служебной процедуры ¤SNDJBC для создания задания. Следовательно, чтобы определить, должен ли быть сформирован регистрационный файл или нет, недостаточно проверить данный код элемента на строку нулевой длины. Необходимо проверить бит QUI¤V_JOB_LOG_NULL кода элемента QUI_JOB_FLAGS;

QJI¤_NOTE -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает в виде строки символов примечание, которое требуется печатать

на страницах флагов задания или файла для указанного задания. Поскольку примечание может включать в себя до 255 символов, то поле "длина буфера" дескриптора элемента должно задавать 255 байтов. Данный код имеет смысл только для выходных исполняемых очередей;

QUI¤_OPERATOR_REQUEST -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает в виде строки символов сообщение, которое должно отсылаться оператору очереди перед тем, как начнет выполняться указанное задание. Поскольку сообщение может включать в себя до 255 символов, то поле "длина буфера" дескриптора элемента должно задавать 255 байтов. Данный код имеет смысл только для выходных исполняемых очередей;

QUI¤_OWNER_UIC -

Если задан данный код элемента (для функции QUI¤_DISPLAY_QUEUE), то программа возвращает идентификатор UIC владельца для указанной очереди в стандартном формате длинного слова;

QUI¤_PAGE_SETUP_MODULES -

Если задан данный код элемента (для функции QUI¤_DISPLAY_FORM), то программа возвращает список имен (через запятую) текстовых модулей, которые должны быть извлечены из библиотеки управления устройствами и отпечатаны на устройстве печати перед печатью каждой страницы указанной формы. Поскольку имя текстового

модуля может включать в себя до 31 символа плюс запятая, отделяющая его от предыдущего имени, то поле "длина буфера" дескриптора элемента должно задавать 32 байта для каждого возможного текстового модуля. Данный код имеет смысл только для выходной исполняемой очереди;

с QUI¤_PARAMETER_1 по QUI¤_PARAMETER_8 -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает строку символов, которая содержит значения параметров, определенных пользователем. В пакетном режиме они составляют значения символов языка DCL с P1 по P8, со значениями параметров, определенных пользователем;

QUI¤_PRIORITY -

Если задан данный код элемента (для функции QUI¤_DISPLAY_JOB), то программа возвращает приоритет указанного задания, который представляет из себя десятичное значение в длинном слове. Спланированный приоритет влияет на тот порядок, в котором инициируются задания, назначенные очереди. Наименьшее значение спланированного приоритета равно 0, наивысшее равно 255, то есть, если очередь содержит задания с планирующим приоритетом 10 и с приоритетом 2, то программа управления очередно сначала инициирует задание с приоритетом 10;

QUI¤_PROCESSOR -

Если задан данный код элемента (для функции

QUI¤_DISPLAY_QUEUE), то программа возвращает имя файла образа симбионта, выполняющего задание на печать, которое инициируется из указанной очереди. Имя файла обычно содержит имя устройства и каталога SYS¤SYSTEM:, а также тип файла EXE. Поскольку имя файла может включать в себя до 39 символов, то поле "длина буфера" дескриптора элемента должно задавать 39 байтов;

QUI¤_PROTECTION -

Если задан данный код элемента (для функции QUI¤_DISPLAY_QUEUE), то программа возвращает в длинном слове маску защиты очереди. Маска защиты очереди показана на рис 68.

Маска защиты очереди

Значение защиты															
Все				Группа				Владелец				Система			
D	E	W	R	D	E	W	R	D	E	W	R	D	E	W	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Значение возможных изменений															
Все				Группа				Владелец				Система			
D	E	W	R	D	E	W	R	D	E	W	R	D	E	W	R
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Рис. 68

Биты с 0 по 15 задают значение защиты: четыре типа доступа (читать, писать, исполнять, удалять), предоставленной четырем классам пользователей (система, пользователь, группа, все). Установленный бит запрещает доступ, а сброшенный бит разрешает.

Биты с 16 по 31 блокируют или разблокируют биты с 0 по 15. Если бит во втором слове установлен, то соответствующий бит в первом слове игнорируется. По умолчанию значение в слове имеет значение (S:E, O:D, G:R, W:W);

QUI¤_FLAGS -

Если задан данный код элемента (для функции QUI¤_DISPLAY_QUEUE), то программа возвращает в длинном слове вектор битов, представляющих варианты обработки, которые должны быть выбраны для указанной очереди. каждому варианту обработки соответствует один бит в векторе. Если программа ¤GETQUI устанавливает бит, то задания, инициируемые из очереди, обрабатываются согласно соответствующему варианту обработки. Каждый бит в векторе имеет символическое имя.

Символические имена (определяются макрокомандой ¤QUIDEF):

1) QUI¤V_QUEUE_BATCH - очередь является пакетной очередью или общей пакетной очередью;

2) QUI¤V_QUEUE_CPU_DEFAULT - лимит времени процессора, принятый по умолчанию, указан для всех заданий в очереди;

3) QUI¤V_QUEUE_CPU_LIMIT - максимальный лимит времени процессора указан для всех заданий в очереди;

4) QUI$V_QUEUE_FILE_BURST - страницы пакета и флагов предшествуют каждому файлу в каждом задании, инициированном из очереди;

5) QUI$V_QUEUE_FILE_BURST_ONE - страница пакета и флагов предшествует только первой копии первого файла в каждом задании, инициированном из очереди;

6) QUI$V_QUEUE_FILE_FLAG - страница флагов предшествует каждому файлу в каждом задании инициируемом из очереди;

7) QUI$V_QUEUE_FILE_FLAG_ONE - страница флагов предшествует только первой копии первого файла в каждом задании, инициированном из очереди;

8) QUI$V_QUEUE_FILE_PAGINATE - выходной симбионт нумерует страницы на выводе для каждого задания, инициированного из данной очереди. Выходной симбионт нумерует страницы, вводя подачу формы всякий раз, когда вывод достигает нижнего края формы;

9) QUI$V_QUEUE_FILE_TRAILER - дополнительная страница следует за каждым файлом каждого задания, инициируемого из очереди;

10) QUI$V_QUEUE_FILE_TRAILER_ONE - дополнительная страница следует только за последней копией последнего файла в каждом задании, инициированном из очереди;

11) QUI$V_QUEUE_GENERIC - очередь является общей очередью;

12) QUI$V_QUEUE_GENERIC_SELECTION - очередь является исполняемой очередью;

13) QUI$V_QUEUE_JOB_BURST - страницы пакета и флагов предшествуют каждому заданию, инициированному из очереди;

14) QUI$V_QUEUE_JOB_FLAG - страница флагов предшествует каждому заданию, инициированному из очереди;

15) QUI$V_QUEUE_JOB_SIZE_SCHEDH - задания, инициированные из очереди, планируются в соответствии с размером. При этом наименьшее задание данного приоритета обрабатывается первым. Имеет смысл только для выходных очередей;

16) QUI$V_QUEUE_JOB_TRAILER - дополнительная страница следует за каждым заданием, инициированным из данной очереди;

17) QUI$V_QUEUE_RECORD_BLOCKING - симбионту разрешено соединять или блокировать вместе выходные записи, которые он посылает на выходное устройство;

18) QUI$V_QUEUE_RETAIN_ALL - все задания, инициированные из очереди, остаются в очереди после того как они закончат выполнение. Завершение задания помечаются кодом состояния завершения;

19) QUI$V_QUEUE_RETAIN_ERROR - в очереди остаются только те задания, которые не завершились успешно;

20) QUI$V_QUEUE_SWAP - можно менять местами задания, инициированные из данной очереди;

21) QUI$V_QUEUE_TERMINAL - очередь является общей очередью, которая может помещать задания только в терминальные очереди;

22) QUI$V_QUEUE_WSDEFAULT - для каждого задания, инициированного из очереди, указан размер рабочего набора,

принятый по умолчанию;

23) QUI$V_QUEUE_WSEXTENT - экстент рабочего набора задан для каждого задания, инициированного из очереди;

24) QUI$V_QUEUE_WSQUATA - для каждого задания, инициированного из очереди, задана квота рабочего набора;

QUI$_QUEUE_NAME -

Если задан данный код элемента (для функций QUI$_DISPLAY_QUEUE, QUI$_DISPLAY_JOB), то программа возвращает в виде строки символов имя указанной очереди или имя очереди, которая содержит указанное задание. Поскольку имя очереди может включать в себя до 31 символа, то поле "длина буфера" дескриптора элемента должно задавать 31 байт;

QUI$_QUEUE_STATUS -

Если задан данный код элемента (для функции QUI$_DISPLAY_QUEUE), то программа возвращает флаги состояния, которые содержатся в длинном слове вектора битов;

Символические имена для данных файлов, которые определены макрокомандой $QUIDEF:

1) QUI$V_QUEUE_ALIGNING - очередь печатает указанное количество выводимой информации, так чтобы можно было правильно выровнять бумагу;

2) QUI$V_QUEUE_IDLE - очередь не содержит запросов на задания;

3) QUI$V_QUEUE_LOWERCASE - очередь связана с устройством печати, которое может печатать как большие так и строч-

ные буквы;

4) QUI$V_QUEUE_PAUSED - выполнение всех текущих заданий в очереди временно приостановлено;

5) QUI$V_QUEUE_PAUSING - очередь временно приостанавливает выполнение. Текущие выполняющиеся задания завершаются; временно ни одно новое задание не начинает выполняться;

6) QUI$V_QUEUE_REMOTE - очередь назначена физическому устройству, которое не связано с локальным узлом;

7) QUI$V_QUEUE_RESETTING - очередь восстановлена заново и остановлена;

8) QUI$V_QUEUE_RESUMING - очередь вновь восстановлена после приостановки;

9) QUI$V_QUEUE_STALLED - физическое устройство, которому назначена очередь, остановлено, то есть данное устройство не закончило обработку последнего направленного к нему запроса на ввод-вывод;

10) QUI$V_QUEUE_STARTING - очередь в состоянии запуска;

11) QUI$V_QUEUE_STOPPED - очередь остановлена;

12) QUI$V_QUEUE_STOPPING - очередь останавливается;

13) QUI$V_QUEUE_UNAVAILABLE - недоступно физическое устройство, которому назначена очередь;

QUI$_REQUEUE_QUEUE_NAME -

Если задан данный код элемента (для функции QUI$_DISPLAY_JOB), то программа возвращает в виде символьной строки имя очереди, которой переназначено указанное задание. Поскольку имя очереди может содержать

до 31 символа, то поле "длина буфера" дескриптора элемента должно задавать до 31 байт;

QUI¤_SEARCH_FLAGS -

Данное значение является значением входного кода элемента, которое указывает длинное слово вектора битов, где каждый бит задает область поиска об'екта, указанного при вызове программы ¤GETQUI. Макрокоманда ¤QUIDEF определяет символические имена для каждого бита в векторе битов. Вектор битов конструируется заданием символических имен для требуемых возможностей в логической операции OR (или). Табл.61 содержит символические имена для каждой возможности и код функции для которого флаг имеет смысл.

Таблица 61

Символическое имя (функциональный код)	Описание
QUI¤V_SEARCH_ALL_JOB (QUI¤_DISPLAY_JOB)	¤GETQUI ищет все задания, включенные в установленный контекст очереди. Если данный флаг не задан, то программа возвращает информацию только о тех заданиях, которые имеют то же имя пользователя, что и вызывающий процесс
QUI¤V_SEARCH_BATCH	ограничивает область поиска для про-

Продолжение табл. 61

Символическое имя (функциональный код)	Описание
(QUI¤S_DISPLAY_QUEUE) QUI¤V_SEARCH_SYMBIONT QUI¤S_DISPLAY_QUEUE)	!граммы только пакетными очередями !ограничивает область поиска только !выходными очередями
QUI¤V_SEARCH_THIS_JOB (QUI¤S_DISPLAY_FILE, QUI¤S_DISPLAY_JOB, QUI¤S_DISPLAY_QUEUE)	!программа возвращает информацию о !вызывающем пакетном задании, выпол- !няющемся в файле команд или об оче- !реди, связанной с вызывающим пакет- !ным заданием. Программа устанавлива- !ет новый контекст очереди и задания !вызывающего процесса. Данный кон- !текст очереди и задания аннулируется !после выполнения программы ¤GETQUI. !Если задан данный флаг, то программа !игнорирует все другие возможности !данного кода элемента
QUI¤V_SEARCH_WILDCARD (QUI¤_DISPLAY_ _CHARACTERISTIC QUI¤_DISPLAY_FORM, QUI¤S_DISPLAY_QUEUE)	!программа выполняет поиск в режиме !случайного запроса, даже если код не !содержит никаких символов, характер- !ных для данного режима

QUI¤_SEARCH_NAME -

Данное значение входного кода элемента задает в виде символьной строки размером от 1 до 31 символа имя об'екта, о котором программа возвращает информацию. В буфере должно быть указано имя характеристики, формы или очереди. Чтобы заставить программу ¤GETQUI выполнять поиск по обобщенным запросам, строка QUI¤_SEARCH_NAME должна содержать один или более символов обобщения (% или *);

QUI¤_SEARCH_NUMBER -

Данное значение входного кода элемента задает в длинном слове десятичное значение, представляющее номер формы или задания, о котором программа возвращает информацию. В буфере должно быть задано десятичное значение размером в длинное слово;

QUI¤_SUBMISSION_TIME -

Если задан данный код элемента, то программа возвращает в квадрослово абсолютное значение времени, отмечающее момент постановки указанного здания в очередь. (Для функции QUI¤_DISPLAY_JOB);

QUI¤_UIC -

Если задан данный код элемента, то программа возвращает в стандартном формате длинного слова идентификатор UIC владельца указанного задания. (Для функции QUI¤_DISPLAY_JOB);

QUI¤_USERNAME -

Если задан данный код элемента, то программа возвращает "имя пользователя" владельца указанного задания.

имя пользователя возвращается в виде 12-байтовой строки с дополняющими нулями. (Для функции QUI¤_DISPLAY_JOB);

QUI¤_WSDEFAULT -

Если задан данный код элемента, то программа возвращает принятый по умолчанию размер рабочего набора для указанного задания или очереди. Значение представляет из себя целое число от 1 до 65535 размером в длинное слово. Данный код имеет смысл только для пакетных заданий и выходных исполняемых очередей. (Для функций QUI¤_DISPLAY_JOB, QUI¤_DISPLAY_QUEUE);

QUI¤_WSEXTENT -

Если задан данный код элемента, то программа возвращает экстент рабочего набора в виде целого числа от 1 до 65535 размером в длинное слово. Данный экстент задается для указанного задания или очереди и имеет смысл только для пакетных заданий и выходных исполняемых очередей. (Для функций QUI¤_DISPLAY_JOB, QUI¤_DISPLAY_QUEUE);

QUI¤_WSQUOTA -

Если задан данный код элемента, то программа возвращает квоту рабочего набора для указанного задания или очереди в виде целого числа от 1 до 65535 размером в длинное слово. Данное значение имеет смысл только для пакетных заданий и выходных исполняемых очередей. (Для функций QUI¤_DISPLAY_JOB, QUI¤_DISPLAY_QUEUE).

IOSB

Использование в МОС ВП: IO_STATUS_BLOCK

Тип: квадрослово (без знака)

Доступ: только запись

Механизм: по ссылке

Блок состояния ввода-вывода, в который программа $GETQUI записывает состояние завершения после окончания запрошенной операции. Аргумент IOSB является адресом блока состояния ввода-вывода.

При инициировании запроса программа устанавливает значение квадрослова блока состояния ввода-вывода равным 0. После завершения запрошенной операции программа записывает значение кода состояния в первое длинное слово блока ввода-вывода. Во второе длинное слово она записывает 0; данное длинное слово не используется.

Хотя данный аргумент не является обязательным, рекомендуется использовать его по следующим причинам:

1) если для сигнализации о завершении программы системного обслуживания используется флаг события, то можно проанализировать в блоке состояния ввода-вывода значение кода состояния, чтобы удостовериться, что данный флаг события не был установлен другим событием;

2) если для синхронизации завершения программы системного обслуживания используется программа $SYNCH, то блок состояния ввода-вывода является обязательным аргументом для данной программы;

3) значения кода состояния, возвращаемые в регистре R0

и в блоке состояния ввода-вывода предоставляют информацию о различных аспектах вызова программы системного обслуживания $GETQUI. Значение кода состояния, возвращаемое в регистре R0, предоставляет информацию об успешном или неуспешном результате вызова программы. Значение же кода состояния, возвращаемое в блоке состояния ввода-вывода, дает информацию об успехе или неудаче операции, выполняемой программой системного обслуживания. Следовательно, для точной оценки результата вызова программы $GETQUI необходимо проверить значения кодов состояния, возвращаемые как в регистре R0 так и блоке состояния ввода-вывода.

ASTADR

Использование в МОС ВП: AST_PROCEDURE

Тип: маска входа в программу

Доступ: вызов без развертывания стека

Механизм: по ссылке

Подпрограмма обработки AST, которая должна выполняться после завершения программы $GETQUI. Аргумент ASTADR является адресом маски входа данной подпрограммы.

Если задан данный аргумент, то подпрограмма AST выполняется в том же режиме доступа, что и процесс, вызвавший программу $GETQUI.

ASTPRM

Использование в МОС ВП: USER_PARM

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Параметр AST, который должен быть передан подпрограмме обработки AST, адрес которой задается аргументом ASTADR. Аргумент ASTPRM представляет длинное слово, содержащее данный параметр.

Описание:

Чтобы получить информацию о задании или файле, пользователь, вызывающий программу ¤GETQUI, должен иметь доступ с чтением или привилегию SYSPRV или OPER. Если пользователь не имеет привилегии для доступа к заданию, указанному в операциях QUI¤_DISPLAY_JOB или QUI¤_DISPLAY_FILE, то программа системного обслуживания ¤GETQUI возвращает успешный код состояния. Однако при этом она устанавливает в коде элемента QUI¤_JOB_STATUS бит QUI¤_JOB_INACCESSIBLE и возвращает информацию только для следующих кодов элементов:

1) QUI¤_AFTER_TIME;

2) QUI¤_COMPLETED_BLOCKS;

3) QUI¤_ENTRY_NUMBER;

4) QUI¤_INTERVENING_BLOCKS;

5) QUI¤_INTERVENING_JOBS;

6) QUI¤_JOB_SIZE;

7) QUI¤_JOB_STATUS.

Программа системного обслуживания ¤GETQUI возвращает информацию об об'ектах, содержащихся в системном файле очередей заданий. В частности, она возвращает информацию о следующих типах об'ектов: характеристики, формы, очереди, задания, содержащиеся в очередях, и файлы, содержащиеся в данных заданиях.

При вызове программы системного обслуживания ¤GETQUI контроллер заданий устанавливает внутренний блок контекста GETQUI (GQC). Операционная система МОС ВП использует блок GQC для временного хранения информации и трассировки последовательности запросов на операции. Система обеспечивает только один блок GQC на процесс, следовательно одновременно может выполняться только одна операция программы ¤GETQUI.

Чтобы дать возможность пользователю получить информацию либо о конкретном об'екте в отдельном вызове программы, либо о нескольких об'ектах в последовательности вызовов, программа ¤GETQUI поддреживает три различных режима поиска. Данные режимы поиска влияют на диспозицию блока GQC:

1) режим поиска по одиночному запросу. Программа ¤GETQUI возвращает информацию о конкретном об'екте в одном вызове. После обработки вызова операционная система МОС ВП аннулирует блок GQC;

2) режим поиска по последовательности запросов. Программа возвращает информацию о нескольких об'ектах одного типа в последовательности вызовов. Операционная система мос вп сохраняет блок GQC между вызовами пока не завершится последовательность запросов;

3) режим поиска по последовательности вложений запросов. В результате последовательности вызовов программа возвращает информацию о заданиях, содрежащихся в выбранном задании. После каждого запроса операционная система МОС ВП сохраняет блок GQC, который может предоставить контекст очереди или задания для последующих вызовов.

В режиме одиночного запроса программа ¤GETQUI может возвращать информацию о следующих об'ектах:

1) конкретном определении характеристики или формы, которые идентифицируются по имени или номеру;

2) конкретном определении по очереди, которая идентифицируется по имени;

3) очереди, которая связана с вызывающим пакетным заданием, самом вызывающем пакетном задании или командном файле программ, который выполняет пакетное задание.

Чтобы получить информацию о конкретном определении характеристики или формы, необходимо задать код функции QUI¤_DISPLAY_CHARACTERISTIC (для характеристики) или QUI¤_DISPLAY_FORM (для формы). Необходимо также задать имя конкретной характеристики или формы в коде элемента QUI¤_SEARCH_NAME или номер данной характеристики или формы в коде элемента QUI¤_SEARCH_NUMBER. Строка имени не должна содержать символов, характерных для режима поиска по последовательности запросов (* или %). Можно задать оба кода элемента QUI¤_SEARCH_NAME и QUI¤_SEARCH_NUMBER, однако данные коды должны быть связаны с одной и той же характеристикой или формой.

Чтобы получить информацию о конкретном определении очереди, следует задать функциональный код QUI¤_DISPLAY_QUEUE и имя очереди указать в коде элемента QUI¤_SEARCH_NAME. Строка имени не должна содержать символов, характерных для режима поиска по последозательности запросов (* или %). Можно ограничить область поиска конк-

ретным типом очереди, если выбрать возможность QUE¤V_SEARCH_BATCH (для поиска пакетных очередей) или QUE¤V_SEARCH_SYMBIONT (для поиска выходных очередей) кода элемента QUE¤_SEARCH_FLAGS. Если же искомая очередь не принадлежит к типу, заданному в коде элемента QUE¤_SEARCH_FLAGS, то программа ¤GETQUI не сможет найти данную очередь.

Программа ¤GETQUI предоставляет возможность, которая позволяет пакетному заданию получать информацию об очереди, задании или файле команд, который выполняет связанное пакетное задание, не используя при этом предварительно режим обобщенного запроса для установки контекста очереди или задания. Можно сделать вызов из пакетного задания, указав код функции QUI¤_DISPLAY_QUEUE, чтобы получить информацию об очереди, из которой инициировано пакетное задание; или код функции QUI¤_DISPLAY_JOB, чтобы получить информацию о самом пакетном задании; или код функции QUI¤_DISPLAY_FILE, чтобы получить информацию о файлах, содержащихся в данном пакетном задании. Для каждого из данных вызовов необходимо выбрать возможность QUI¤V_SEARCH_THIS_JOB кода QUI¤_DISPLAY_FLAGS. Если указана данная возможность, то программа игнорирует все другие возможности в элементом коде.

В режиме поиска по последовательности запросов операционная система МОС ВП сохраняет блоки GQC между вызовами программы ¤GETQUI так что можно выполнить последовательность вызовов программы, чтобы получить информацию обо всех

характеристиках, формах или очередях, содержащихся в системном файле очередей заданий.

Чтобы установить режим поиска по последовательности запросов для определения характеристики или формы, необходимо задавать функциональный код QUI¤_DISPLAY_CHARACTERISTIC (для характеристик) или QUI¤_DISPLAY_FORM (для форм), а в элементном коде QUI¤_SEARCH_NAME имя, которое включает один или более символов (* или %) характеристик для данного режима поиска. Чтобы ограничить область поиска, можно также указать тип очереди, задав возможность QUI¤V_SEARCH_BATCH или QUI¤V_SEARCH_SYMBIONT кода элемента QUI¤_SEARCH_FLAGS. Если задана первая из данных возможностей, то программа возвращает информацию о пакетных очередях, если же задана вторая возможность, то программа возвращает информацию о выходных очередях.

Можно также запустить режим поиска по последовательности запросов для операций вывода на дисплей информации о характеристиках, формах или очередях, если в коде элемента QUI¤_SEARCH_FLAGS указать возможность QUI¤V_SEARCH_WILDCARD. В данном случае операционная система MOC ВП сохраняет блок GQC между вызовами программы, даже если имя, указанное в коде элемента QUI¤_SEARCH_NAME, не включает символов (* или %), характерных для режима поиска по последовательности запросов. Такой метод запуска данного режима полезен, если вызов программы ¤GETQUI происходит в выход из которого осуществляется при возврате значения кода

состояния, которое свидетельствует о неудачном завершении операции. Независимо от содержания имени QUI¤V_SEARCH_WILDCARD гарантирует включение режима поиска по последовательности запросов.

После того, как включен режим поиска по последовательности запросов, он работает до тех пор, пока не произойдет одно из следующих событий, что приведет к аннулированию блока GQC:

1) программа ¤GETQUI по запросу на вывод информации об очереди, характеристики или формы возвращает значение кода состояния JBC¤_NOMOREXXX или ¤JBC_NOSUCHXXX, где "XXX" - CHAR, FORM или QUE;

2) пользователь явно отменяет операцию поиска по последовательности запросов, указав при вызове программы ¤GETQUI функциональный код QUI¤_CANCEL_OPERATION;

3) процесс завершается.

В режиме поиска по последовательности вложенных запросов операционная система МОС ЭП сохраняет блок GQC между вызовами программы ¤GETQUI, так что можно выполнять последовательность вызовов данной программы с целью получения информации о заданиях, которые содержатся в выбранной очереди, или о файлах выбранных заданий. Две из операций программы ¤GETQUI: QUI¤_DISPLAY_JOB и QUI¤_DISPLAY_FILE, могут выполняться только в режиме поиска по последовательности вложенных запросов, с одним исключением. Исключение касается вызова программы ¤GETQUI из пакетного задания для выполнения двух данных операций.

Программа ¤GETQUI нуждается в режиме поиска по последовательности вложенных запросов из-за способа, которым программа отыскивает об'екты и взаимосвязи между очередями, заданиями и файлами. Программа ¤GETQUI выполняет одну операцию на один вызов. Таким образом, она может находить и возвращать информацию только об одном об'екте за один вызов. Однако очереди являются об'ектами, которые содержат задания, а задания, в свою очередь, являются об'ектами, которые содержат файлы. Следовательно, чтобы получить информацию об об'екте, который содержится в другом об'екте, необходимо сначала сделать вызов программы ¤GETQUI, который задает и локализует содержащий об'ект, а затем сделать вызов, чтобы запросить информацию о содержимом об'екте. Операционная система МОС ВП сохраняет информацию о местоположении содержащего об'екта в блоке GQC.

Например, необходимо сделать по крайней мере два вызова программы ¤GETQUI, перед тем, как сделать запрос информации о файле. В первой последовательности вызовов задается операция QUI¤_DISPLAY_QUEUE. Первая последовательность продолжается до тех пор, пока не будет найдена очередь, содержащая задание, содержащее нужный файл. Данный вызов устанавливает контекст очереди. Во второй последовательности вызовов задается операция QUI¤_DISPLAY_JOB. Данные вызовы просматривают очередь, найденную в первой последовательности вызовов, пока не будет найдено задание, содержащее нужный файл. Данный вызов устанавливает контекст задания. Теперь, найдя очередь и задание, содержащее нужный файл,

можно использовать операцию QUI¤_DISPLAY_FILE, чтобы запросить информацию о файле.

Чтобы включить режим поиска по последовательности вложенных запросов для получения информации о задании или файле, необходимо выполнить программу ¤GETQUI в режиме поиска по последовательности запросов, что позволит установить контекст очереди, необходимой для вложенных запросов на операции информации о задании или файле. Включить режим поиска по последовательности запросов для операций вывода информации об очереди можно двумя способами: либо указать имя, содержащее характерные для данного режима символы (* или %) в коде элемента QUI¤_SEARCH_NAME; либо указав обычное имя очереди и выбрав возможность QUI¤V_SEARCH_WILDCARD кода элемента QUI¤_SEARCH_FLAGS. Второй метод ввода режима поиска по последовательности запросов полезен, если пользователю требуется получить информацию об одном или более заданиях или файлах внутри заданий для указанной очереди и, следовательно, требуется указать обычное имя очереди, но при этом необходимо все же сохранить блок GQC после того, как установлен контекст очереди.

Если вызов программы ¤GETQUI делается с кодом функции QUI¤_DISPLAY_JOB, то по умолчанию программа находит те задания в выбранной очереди, которые имеют то же "имя пользователя", что и вызывающий процесс. Если же необходимо получить информацию обо всех заданиях в выбранной очереди, то необходимо выбирать возможности QUI¤V_SEARCH_ALL_JOBS кода элемента QUI¤_SEARCH_FLAGS.

После того, как установлен контекст очереди, он сохраняется до тех пор, пока либо пользователь не изменит его, вызвав еще раз программу $GETQUI с кодом функции QUI$_DISPLAY_QUEUE, либо не будет аннулирован блок GQC. Установленный контекст очереди сохраняется, пока либо пользователь не изменит контекст, сделав еще раз вызов программы $GETQUI с кодом функции QUI$_DISPLAY_JOB, либо программа не вернет код состояния JBC$_NOMOREJOB или JBC$_NOSUCHJOB. Хотя возврат любого из данных двух значений кода состояния освобождает контекст очереди, режим поиска по последовательности запросов остается включенным, поскольку блок GQC продолжает сопровождать контекст очереди.

Возвращаемые значения кода состояния:

SS$_NORMAL - успешное завершение;

SS$_ACCVIO - вызывающая программа не может прочитать список элементов или входной буфер; или не может записать в буфер возвращаемой длины, выходной буфер или блок состояния;

SS$_BADPARAM - неверный код функции. Список элементов содержит неверный код элемента. Дескриптор буфера имеет неверную длину. Зарезервированный параметр имеет ненулевое значение;

SS$_DEVOFFLINE - не выполняется процесс контроллера заданий;

SS$_EXASTLM - указан аргумент ASTADR и процесс

превысил квоту ASTLM;

SS¤_ILLEFC - аргумент EFN указывает неверный номер флага события;

SS¤_INSFMEM - стек режима управляющей программы содержит недостаточную память для выполнения запроса;

SS¤_MBFULL - почтовый ящик контроллера заданий переполнен;

SS¤_MBTOOSML - сообщение, посланное в почтовый ящик, слишком велико для почтового ящика контроллера заданий;

SS¤_UNASEFC - аргумент EFN указывает на неподсоединенный кластер флагов событий.

Значения кода состояния, возвращаемые в блок состояния ввода-вывода:

JBC¤_NORMAL - нормальное завершение;

JBC¤_INVFUNCOD - указан неверный код функции;

JBC¤_INVITMCOD - список элементов содержит неверный код элемента;

JBC¤_INVPARLEN - длина заданной строки находится вне допустимого диапазона для данного кода элемента;

JBC¤_INVQUENAM - имя очереди синтаксически неверно;

JBC¤_JOBQUEDIS - запрос не может быть обработан, поскольку не была запущена системная программа;

JBC¤_MISREQPAR - не задан код элемента, который тре-

буется для указанного кода функции;

JBC¤_NOJOBCTX - для операции QUI¤_DISPLAY_FILE не был установлен контекст задания;

JBC¤_NOMORECHAR - больше нет заданных характеристик. Отмечает завершение операции QUI¤_DISPLAY_CHARACTERISTIC в режиме поиска по последовательности запросов;

JBC¤_NOMOREFILE - больше нет файлов, связанных с текущим контекстом заданий. Отмечает конец операции QUI¤_DISPLAY_FILE в режиме поиска по последоватльности запросов для текущего контекста заданий;

JBC¤_NOMOREFORM - больше нет заданных форм. Отмечает завершение операции QUI¤_DISPLAY_FORM в режиме поиска по последовательности запросов;

JBC¤_NOMOREJOB - больше нет заданий, связанных с текущим контекстом заданий. Отмечает конец операции QUI¤_DISPLAY_JOB в режиме поиска по последовательности запросов для текущего контекста заданий;

JBC¤_NOMOREQUE - больше нет заданных очередей. Отмечает конец операции QUI¤_DISPLAY_QUEUE в режиме поиска по

последовательности запросов;

JBC¤_NOQUECTX - для операции QUI¤_DISPLAY_JOB или QUI¤_DISPLAY_QUEUE не был устэновлен контекст очереди;

JBC¤_NOSUCHCHAR - указанная характеристика отсутствует;

JBC¤_NOSUCHFILE - указанный файл отсутствует;

JBC¤_NOSUCHFORM - указанная форма отсутствует;

JBC¤_NOSUCHJOB - указанное задание отсутствует;

JBC¤_NOSUCHQUE указанная очередь отсутствует.

13.64. ¤GETQUIW - получить информацию об очереди и ждать завершения

Программа системного обслуживания ¤GETQUIW возвращает информацию об очередях и заданиях, инициированных из данных очередей. Программа системного обслуживания ¤SNDJBC является основным интерфейсом с контроллером заданий операционной системы МОС ВП, которой является программой учета и управления очередями МОС ВП.

Программа системного обслуживания ¤GETQUIW завершается синхронно, то есть она возвращает управление вызывающей программе вместе с запрошенной информацией. Для асинхронного завершения необходимо использовать программу системного обслуживания "получить информацию об очереди" (¤GETQUI). Данная программа возвращает управление вызывающей программе поставив в очередь запрос на информацию и не ожидая при этом возврата данной информации.

Во всех остальных отношениях программы $GETQUIW и $GETQUI идентичны. Всю информацию о программе $GETQUIW можно получить в описании программы $GETQUI. Дополнительная информация о завершении программ системного обслуживания приведена в описании программы "синхронизировать" ($SYNCH).

Формат:

SYS$GETQUIW [EFN],FUNC[,NULLARG][,ITMLST][,IOSB]
[,ASTADR][,ASTPRM]

13.65. $GETSYI - получить информацию о системе

Программа системного обслуживания $GETSYI возвращает информацию о локальной системе МОС ВП.

Программа $GETSYI завершается синхронно, то есть возвращает управление вызывающей программе вместе с запрошенной информацией. Для синхронного завершения рекомендуется использовать программу системного обслуживания $GETSYIW.

Дополнительная информация о завершении программы системного обслуживания приведена в подразделе 2.5 и в описании программы системного обслуживания "синхронизировать" ($SYNCH).

Формат:

SYS$GETSYI [EFN],[CSIDADR],[NODENAME],ITMLST
,[IOSB],[ASTADR],[ASTPRM]

Аргументы:

EFN

Использование в МОС ВП: EF_NUMBER

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Номер флага события, который должен быть установлен, когда программа $GETSYI обрабатывает запрос. Аргумент EFN является длинным словом, содержащим данный номер.

После инициирования запроса программа $GETSYI сбрасывает указанный флаг события (или флаг события с номером 0, если аргумент EFN не задан). Затем после завершения обработки запроса устанавливается указанный флаг события (или флаг события с номером 0).

CSIDADR

Использование в МОС ВП: PROCESS_ID

Тип: длинное слово (без знака)

Доступ: модификация

Механизм: по ссылке

Данный аргумент не используется.

NODENAME

Использование в МОС ВП: PROCESS_NAME

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Данный аргумент не используется.

ITMLST

Использование в МОС ВП: ITEM_LIST_3

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Список элементов, указывающий, какая информация о системе должна быть возвращена. Аргумент ITMLST является адресом списка дескрипторов, каждый из которых описывает элемент информации. Данный список дескрипторов элементов завершается длинным словом, содержащим значение 0. Рис.69 описывает отдельный дескриптор элемента.

Дескриптор элемента

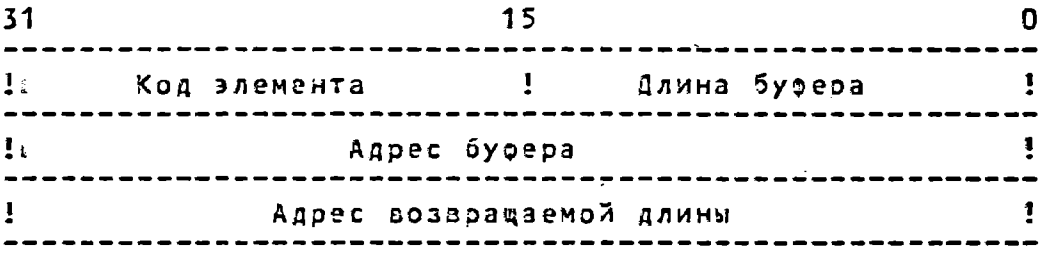


Рис. 69

Поля дескриптора элемента программы ¤GETSYI:

длина буфера -

Слово, содержащее предоставленное пользователем целое число, указывающее длину (в байтах) буфера, в который программа должна записывать информацию. Необходимая длина буфера зависит от кода элемента, заданного в поле "код элемента" дескриптора элемента. Если значение поля "длина буфера" слишком мало, программа ¤GETSYI усекает данные;

код элемента -

Слово, содержащее предоставленный пользователем символический код, указывающий элемент информации, который должна вернуть программа. Данные коды определяются макрокомандой ¤SYIDEF;

адрес буфера -

Длинное слово, содержащее предоставленный пользователем адрес буфера, в который программа должна записывать информацию;

адрес возвращаемой длины -

Длинное слово, содержащее предоставленный пользователем адрес слова, в который программа ¤GETSYI записывает реальную длину в байтах возвращаемой информации;

коды элементов программы ¤GETSYI:

SYI¤_BOOTTIME -

Если задан данный код элемента, то программа возвращает время загрузки системы. Поскольку возвращаемое время имеет стандартный 64-битовый формат абсолютного времени, то поле "длина буфера" дескриптора элемента должно задавать 8 байтов;

SYI¤_CHARACTER_EMULATED -

Если задан данный код элемента, то программа возвращает число 1, если инструкции символьных строк эмулированы на центральном процессоре, и 0, если нет. Поскольку данное число является булевым значением, (1 или 0), то поле "длина буфера" в дескрипторе элемента должно задавать 1 байт;

SYI¤_DECIMAL_EMULATED -

Если задан данный код элемента, то программа возвращает число 1, если операции с десятичными строками эмулированы на центральном процессоре, и 0, если нет. Поскольку данное число является булевым значением (1 или 0), то поле "длина буфера" дескриптора элемента должно задавать 1 байт;

SYI¤_D_FLOAT_EMULATED -

Если задан данный код элемента, то программа возвращает число 1, если операции процессоре, и 0, если нет. Поскольку данное число является булевым значением (1 или 0), то поле "длина буфера" дескриптора элемента должно задавать 1 байт.

SYI¤_F_FLOAT_EMULATED -

Если задан данный код элемента, то программа возвращает число 1, если операции над числами с плавающей запятой F_формата эмулированы на центральном процессоре, и 0, если нет. Поскольку данное число является булевым значением (1 или 0), то поле "длина буфера" дескриптора элемента должно задавать 1 байт;

SYI¤_G_FLOAT_EMULATED -

Если задан данный код элемента, то программа возвращает число 1, если операции над числами с плавающей запятой G_формата эмулированы на центральном процессоре, и 0, если нет. Поскольку данное число является булевым значением (1 или 0), то поле "длина буфера" дескриптора элемента должно задавать 1 байт;

SYI¤_H_FLOAT_EMULATED -

Если задан данный код элемента, то программа возвращает число 1, если операции над числами с плавающей запятой H_формата эмулированы на центральном процессоре, и 0, если нет. Поскольку данное число является булевым значением (1 или 0), то поле "длина буфера" дескриптора элемента должно задавать 1 байт;

SYI¤_NODE_AREA -

Если задан данный код элемента, то программа возвращает область DECNET узла. Поскольку область DECNET представляет из себя десятичное число размером в длинное слово, то поле "длина буфера" дескриптора элемента должно задавать 4 байта;

SYI¤NODE_NUMBER -

Если задан данный код элемента, то программа возвращает номер DECNET данного узла. Поскольку номер DECNET является десятичным числом размером в длинное слово, то поле "длина буфера" дескриптора элемента должно задавать 4 байта;

SYI¤PAGEFILE_FREE -

Если задан данный код элемента, то программа возвращает число свободных страниц в текущих страничных файлах. Поскольку данное число имеет размер длинного слова, то поле "длина буфера" дескриптора должна задавать 4 байта;

SYI¤PAGEFILE_PAGE -

Если задан данный код элемента, то программа возвра-

щает число страниц в текущих страничных файлах. Поскольку данное число имеет размер длинного слова, то поле "длина буфера" дескриптора элемента должно задавать 4 байта;

SYI$_SID -

Если задан данный код элемента, то программа возвращает содержание регистра системного идентификатора. Поскольку значение данного регистра представляет из себя шестнадцатиричное число размером в длинное слово, то поле "длина буфера" дескриптора элемента должно задавать 4 байта;

SYI$SWAPFILE_FREE -

Если задан данный код элемента, то программа возвращает число свободных страниц в текущих свопинговых файлах. Поскольку данное число имеет размер длинного слова, то поле "длина буфера" дескриптора элемента должно задавать 4 байта;

SYI$SWAPFILE_PAGE -

Если задан данный код элемента, то программа возвращает число страниц в текущих свопинговых файлах. Поскольку данное число имеет размер длинного слова, то поле "длина буфера" дескриптора элемента должно задавать 4 байта;

SYI$_VERSION -

Если задан данный код элемента, то программа возвращает в виде строки символов номер версии программного обеспечения для операционной системы МОС ВП. Поскольку

Номер версии является 8-байтовой строкой с дополняющими пробелами, то поле "длина буфера" дескриптора элемента должно задавать 4 байта;

SYI$_XXXX -

Если задан данный код элемента, то программа возвращает текущее значение параметра генерации SYSGEN с именем "XXXX". Буфер должен иметь размер длинного слова, в котором программа записывает значение указанного параметра генерации SYSGEN.

IOBS

Использование в МОС ВП: IO_STATUS_BLOCK

Тип: квадрослово (без знака)

Доступ: только запись

Механизм: по ссылке

Блок состояния ввода-вывода, в который записывается конечное состояние завершения. Аргумент IOBS является адресом квадрослова, содержащего блок состояния ввода-вывода.

Если аргумент IOSB задан, то программа $GETSYI при инициировании запроса заполняет данное квадрослово нулевым значением. После завершения обработки запроса данное значение кода состояния записывается в первое длинное слово, а второе длинное слово зарезервировано.

Хотя данный аргумент не является обязательным, рекомендуется задавать его по следующим причинам:

1) если для сигнализации о завершении программы системного обслуживания используется флаг событий, то можно проверить в блоке состояния ввода-вывода значение кода сос-

тояния, чтобы удостовериться, что флаг состояния не был установлен каким-либо другим событием, отличным от завершения программы;

2) если для синхронизации завершения программы системного обслуживания используется программа ¤SYNCH, то блок состояния ввода-вывода является обязательным аргументом для данной программы;

3) значение кода состояния, возвращаемое в регистре R0 и значение кода состояния возвращаемое в блоке состояния ввода-вывода, дают информацию о различных аспектах вызова программы системного обслуживания ¤GETSYI. Значение кода состояния, возвращаемое в регистре R0, дает информацию об успешном или неудачном вызове программы, а значение кода состояния, возвращаемое в блоке состояния ввода-вывода, дает информацию об успешном или неудачном выполнении операции программы системного обслуживания. Таким образом, чтобы получить точную информацию о результате обращения к программе ¤GETSYI, необходимо проверить значения кодов состояния, возвращаемые как в регистре R0, так и в блоке состояния ввода-вывода.

ASTADR

Использование в МОС ВП: AST_PROCEDURE

Тип: маска входа программы

Доступ: вызов без разгрузки стека

Механизм: по ссылке

Подпрограмма обработки AST, которая должна выполняться после завершения программы ¤GETSYI. Аргумент ASTADR являет-

ся адресом маски входа данной подпрограммы.

Если задан аргумент ASTADR, то подпрограмма обработки будет выполняться с тем же режимом доступа, что и процесс, вызвавший программу системного обслуживания $GETSYI.

ASTPRM

Использование в МОС ВП: USER_ARG

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Параметр AST, который должен передаваться подпрограмме обработки AST, заданной аргументом ASTADR. Аргументы ASTPRM представляют из себя длинное слово, содержащее данный параметр.

Описание

Данная программа системного обслуживания использует квоту лимита AST процесса (ASTLM).

Возвращаемые значения кода состояния:

SS$_NORMAL - программа системного обслуживания успешно завершилась;

SS_BADPARAM - список элементов содержит неверный код элемента;

SS_ACCVIO - вызывающая программа не может прочитать список элементов, или не может записать в буфер, заданный полем "адрес буфера" дескриптора элемента, или не может записать в поле "адрес возвращаемой длины" дескриптора элемента;

SS_EXASTLM - процесс превысил свою квоту лимита AST.

Значения кода состояния, возвращаемые в блоке состояния ввода-вывода: совпадают со значениями, возвращаемыми в регистре R0.

13.66. $GETSYIW - получить информацию о системе и ждать

Программа системного обслуживания $GETSYIW возвращает информацию о системе МОС ВП.

Программа системного обслуживания $GETSYIW подобна программе системного обслуживания "получить информацию о системе" ($GETSYI). Программы $GETSYIW и $GETSYI завершаются синхронно, то есть они возвращают управление вызывающей программе вместе с запрошенной информацией.

Всю остальную информацию о программе системного обслуживания $GETSYIW можно получить из описания программы $GETSYI.

Дополнительная информация о завершении программ системного обслуживания приведена в описании программы системного обслуживания "синхронизировать" ($SYNCH) и в подразделе 2.5.

Формат:

SYS$$GETSYIW [EFN],[CSIDADR],[NODENAME],ITMLST
[,IOSB][,ASTADR][,ASTPRM]

13.67. ¤GETTIM - получить время

Программа системного обслуживания ¤GETTIM возвращает текущее системное время в 64-битовом формате.

Формат:

SYS¤GETTIM TIMADR

Аргумент:

TIMADR

Использование в МОС ВП: DATE_TIME

Тип: квадрослово (без знака)

Доступ: только запись

Механизм: по ссылке

Адрес квадрослова, в которое требуется записать текущее время в 64-битовом формате.

Описание

Системное время обновляется через каждые 10 миллисекунд, и возвращаемое время измеряется в 100-наносекундных единицах от системного базового времени.

Дополнительная информация о системном времени содержится в разделе 9.

Возвращаемые значения кода состояния:

SS¤_NORMAL - успешное завершение;

SS¤_ACCVIO - вызывающая программа не может записать квадрослово, предназначенное для получения времени.

13.68. ¤GRANTID - назначить идентификатор процессу

Программа системного обслуживания ¤GRANTID добавляет запись с указанным идентификатором к списку прав процесса или к системному списку прав. Если данный идентификатор уже есть в списке прав, то модифицируются его атрибуты в соответствии с заданием.

Формат:

SYS¤GRANTID [PIDADR],[PRCNAM],[ID],[NAME]
,[PRVATR]

Аргументы:

PIDADR

Использование в МОС ВП: PROCESS_ID

Тип: длинное слово (без знака)

Доступ: модификация

Механизм: по ссылке

Идентификатор процесса (PID), с которым работает программа ¤GRANTID. Аргумент PIDADR является адресом длинного слова, содержащего PID данного процесса. Для идентификации системного списка прав необходимо использовать минус 1. Когда передается аргумент PIDADR, он также и возвращается, поэтому его необходимо передавать не как константу, а как переменную. Если не указан ни аргумент PIDADR, ни аргумент PRCNAM, то используется процесс пользователя, вызвавший программу.

PRCNAM

Использование в МОС ВП: PROCESS_NAME

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Имя процесса, с которым работает программа $GRANTID. Аргумент PRCNAM является адресом дескриптора строки символов, содержащей имя процесса. Максимальная строка имени равна 15 символам. Поскольку номер группы UIC интерпретируется как часть имени процесса, то необходимо использовать аргумент PIDADR, чтобы задавать список прав процесса в другой группе. Если не задан ни один из аргументов PIDADR или PRCNAM, то используется процесс пользователя, вызвавший программу.

ID

Использование в МОС ВП: RIGHTS_HOLDER

Тип: квадрослово (без знака)

Доступ: модификация

Механизм: по ссылке

Идентификатор и атрибуты, которые назначаются процессу при завершении программы $GRANTID. Аргумент является адресом квадрослова, содержащего в первом длинном слове двоичный код идентификатора, который должен быть назначен процессу, а во втором длинном слове - атрибуты.

Символическое значение атрибутов определяется в системной макробиблиотеке (макрокоманда ¤KGBDEF):

- KGB¤V_RESOURCE - позволяет держателю затребовать для идентификатора ресурсы типа блоков на диске.

Символические значения являются смещениями в битах внутри длинного слова. Данные значения можно также получить в виде маски с установленным соответствующим битом, используя вместо префикса KGB¤V префикс KGB¤M.

Один из аргументов ID или NAME должен быть задан. Поскольку аргумент ID возвращается, если указан аргумент NAME или передается он сам, то необходимо задавать его не в виде константы, а в виде переменной.

NAME

Использование в МОС ВП: CHAR_STRING

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Имя идентификатора, назначаемого в результате работы программы ¤GRANTID. Аргумент NAME является адресом дескриптора, указывающего на имя идентификатора. Один из аргументов ID или NAME должен быть задан.

PRVATR

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Предыдущие атрибуты идентификатора. Аргумент PRVATR является адресом длинного слова, используемого для хранения атрибутов идентификатора, если он ранее был представлен в списке прав. Если же идентификатор добавляется, а не модифицируется, то данный параметр игнорируется.

Описание

Программа системного обслуживания "назначить идентификатор процессу" добавляет указанный идентификатор к списку прав процесса или к системному списку прав. Если данный идентификатор уже имеется в списке прав, то его атрибуты меняются на те, которые заданы. Данная программа системного обслуживания предназначена для использования привилегированной подсистемой для изменения профиля прав доступа пользователя по установочному принципу. Она не предназначена для использования обычным пользователем системы.

Для вызова данной программы системного обслуживания требуется привилегия CMKRNL. Кроме того, для модификации списка прав процесса в той же группе, что и вызывающий процесс (если процесс не имеет тот же идентификатор, что и вызывающий процесс), требуется привилегия GROUP. Для модификации списка прав процесса вне группы вызывающего процесса требуется привилегия WORLD. Для модификации системного списка прав требуется привилеия SYSNAM.

В табл.62 сведены результаты передачи аргументов PIDADR и PRCNAM в различных сочетаниях.

Таблица 62

PRCAAM	RIDADR	Результат
олущен	опущен	используется текущий идентификатор процесса. Идентификатор процесса не возвращается
олущен	0	используется текущий идентификатор процесса. Идентификатор процесса возвращается
олущен	задан	используется заданный идентификатор процесса. Идентификатор процесса возвращается
задан	опущен	используется заданное имя процесса. Идентификатор процесса не возвращается
задан	0	используется заданное имя процесса. Идентификатор процесса возвращается
задан	задан	используется заданный идентификатор процесса. Идентификатор процесса возвращается. Имя процесса игнорируется

В табл.63 сведены результаты передачи аргументов ID и NAME в различных сочетаниях.

Таблица 63

NAME	ID	Результат
опущен	опущен	недопустимое сочетание
опущен	задан	используется заданное значение идентификатора. Значение идентификатора возвращается

Продолжение табл. 63

NAME	ID	Результат
задан	опущен	используется заданное имя идентификатора. Значение идентификатора не возвращается
задан	0	используется заданное имя идентификатора. Значение идентификатора возвращается
задан	задан	используется заданное значение идентификатора. Значение идентификатора возвращается, имя идентификатора игнорируется

Возвращаемые значения кода состояния:

SS$_WASCLR - программа системного обслуживания успешно завершилась. Список прав не содержал указанного идентификатора;

SS$_WASSET - программа системного обслуживания успешно завершилась. Список прав уже содержит указанный идентификатор;

SS$_ACCVIO - невозможно прочитать или записать аргумент PIDADR; или невозможно прочитать аргумент PRCNAM; или невозможно прочитать или записать аргумент ID; или невозможно прочитать аргумент NAME; или невозможно записать аргумент PRVATR;

SS$_IVIDENT - заданный идентификатор или держатель имеет неверный формат. Или заданные

идентификатор и держатель совпадают;

SS$_INSFMEM - для открытия базы данных прав нет доступной динамической памяти процесса достаточного размера;

SS$_NOPRIV - вызывающий процесс не имеет привилегии CMKRNL; или выполняется не в режиме управления или ядра; или вызывающий процесс не имеет требуемой привилегии GROUP, WORLD или SYSNAM;

SS$_NOSUCHID - заданное имя идентификатора отсутствует в базе данных прав. Если задан двоичный идентификатор, то он не проверяется в базе данных прав;

SS$_RIGHTSFULL - список прав процесса или системы заполнен;

SS$_NOSYSNAM - операция требует привилегию SYSNAM;

SS$_IVLOGNAM - неверно задано логическое имя;

SS$_NONEXPR - задан несуществующий процесс;

SS$_PRV - пользователь не имеет доступа с чтением в базе данных прав.

Поскольку база данных прав является индексным файлом, доступ к которому осуществляется с помощью системы управления данными СУД-32, то данная программа системного обслуживания может также возвращать коды состояния СУД-32, связанные с операциями над индексными файлами (см. документ [1]).

13.69. ¤HIBER - перевести в состояние спячки

Программа системного обслуживания ¤HIBER позволяет процессу перевести себя в неактивное состояние, но остаться при этом известным системе, так что он может быть прерван, например, для получения AST. Запрос на спячку - это запрос "ждать пробуждения". Когда вызывается программа системного обслуживания "пробудить процесс от спячки" (¤WAKE), или когда истекло время, заданное программой системного обслуживания "запланировать пробуждение" (¤SCHDWK), процесс продолжает выполнение с инструкции, следующей за вызовом программы ¤HIBER.

Формат:

SYS¤HIBER

Аргументы:

Нет.

Описание:

В языке макроассемблер программу системного обслуживания ¤HIBER можно вызвать только при помощи макрокоманды ¤HIBER_S.

Процесс, находящийся в состоянии спячки, можно выгрузить из балансного набора, если данный процесс не зафиксирован в балансном наборе.

Состояние ожидания, вызванное программой ¤HIBER, может быть прервано посредством AST, если режим доступа, в котором выполняется AST, имеет равную или высшую привилегию по сравнению с режимом доступа, для которого был выдан запрос на спячку, и процесс допускает AST данного режима доступа.

Когда программа обработки AST завершает выполнение, система снова выполняет программу системного обслуживания ¤HIBER для данного процесса. Если во время выполнения программы обработки AST был выдан запрос на пробуждение (данным же или другим процессом), то процесс возобновляет выполнение. Если запрос на пробуждение не выдавался, то процесс продолжает спячку.

Если для процесса, который не находился в состоянии спячки, выдается один или несколько запросов на пробуждение, то следующий вызов программы ¤HIBER сразу же возвращает управление, то есть, процесс не переходит в состояние спячки. Для выставленных запросов на пробуждение счетчик не ведется.

Хотя данная программа не имеет аргументов, в обращении к функции на языке фортран необходимо использовать скобки, чтобы указать нулевой список аргументов, как в следующем примере:

```
ISTAT=SYS¤HIBER()
```

Возвращаемое значение кода состояния:

SS¤_NORMAL - успешное завершение.

13.70. ¤IDTOASC - транслировать идентификатор в имя идентификатора

Программа системного обслуживания ¤IDTOASC транслирует заданное значение идентификатора в имя данного идентификатора.

Формат:

SYS$IDTOASC ID,[NAMLEN],[NAMBUF],[RESID],[ATTRIB]
,[CONTXT]

Аргументы:

ID

Использование в МОС ВП: RIGHTS_ID

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Двоичное значение идентификатора, транслируемое программой $IDTOASC. Аргумент ID является длинным словом, содержащим данное двоичное значение идентификатора. Чтобы получить все имена идентификаторов в базе данных прав следует задать аргумент ID=-1 и вызывать программу $IDTOASC в цикле до тех пор, пока не будет получен код состояния SS$_NOSUCHID. Идентификаторы возвращаются в алфавитном порядке.

NAMLEN

Использование в МОС ВП: WORD_UNSIGNED

Тип: слово (без знака)

Доступ: только запись

Механизм: по ссылке

Число символов в имени идентификатора, полученном в результате трансляции программой $IDTOASC. Аргумент NAMLEN является адресом слова, содержащего длину имени идентификатора, которое записывается в аргумент NAMBUF.

NAMBUF

Использование в МОС ВП: CHAR_STRING

Тип: текстовая строка

Доступ: только запись

Механизм: по дескриптору - дескриптор строки фиксированной длины

Текстовая строка имени идентификатора, которая возвращается в результате трансляции программой $IDTOASC. Аргумент NAMBUF является адресом дескриптора, указывающего на буфер, в который записывается имя идентификатора.

RESID

Использование в МОС ВП: RIGHTS_ID

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Значение идентификатора имени идентификатора, возвращаемого в аргументе NAMBUF. Аргумент RESID является адресом длинного слова, содержащего 32-битовый код идентификатора.

ATTRIB (атрибуты)

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Маска атрибутов, связанных с идентификатором, возвращаемым в аргументе RESID. Аргумент ATTRIB является адресом длинного слова, содержащего маску атрибутов.

В системной макробиблиотеке (макрокоманда ¤KGBDEF) определяется символическое значение:

1) KGB¤V_RESOURCE - позволяет держателю запрашивать для идентификатора ресурсы типа блоков на дисках.

Символическое значение является смещением в битах внутри длинного слова. Данное значение можно также получить в виде маски, если вместо префикса KGB¤V использовать префикс KGB¤M.

CONTXT

Использование в МОС ВП: CONTEXT

Тип: длинное слово (без знака)

Доступ: модификация

Механизм: по ссылке

Значение контекста, используемое при повторяющихся вызовах программы ¤IDTOASC. Аргумент CONTXT является адресом длинного слова, используемого при поиске всех идентификаторов. Значение контекста первоначально должно быть равным нулю, а результирующий контекст каждого вызова программы ¤IDTOASC должен быть представлен для каждого последующего вызова. После того, как аргумент CONTXT передан программе ¤IDTOASC, его нельзя модифицировать.

Описание

Програма системного обслуживания "транслировать идентификатор в имя идентификатора" транслирует указанный двоичный идентификатор в имя идентификатора, она может быть также использована для нахождения всех идентификаторов в базе данных прав. Чтобы определить все идентификаторы, сле-

дует вызывать программу $IDTOASC повторно до тех пор, пока она не вернет код состояния SS$_NOSUCHID. То, что возвращается данный код состояния, означает, что программа вернула все идентификаторы, очистила значение контекста и перераспределила цепочку записей.

Чтобы прекратить вызовы программы $IDTOASC до того, как она вернет код состояния SS$_NOSUCHID, необходимо использовать программу SYS$FINISH_RDB для очистки контекста и перераспределения цепочки записей.

Если данная программа системного обслуживания используется в режиме поиска с символами обобщения, то записи возвращаются в порядке имен идентификаторов.

Возвращаемые значения кода состояния:

SS$_NORMAL - программа системного обслуживания нормально завершилась;

SS$_ACCVIO - вызывающий процесс не может записать аргументы NAMLEN, NAMBUF, RESID, ATTRIB или CONTXT;

SS$_INSFMEM - для открытия базы данных прав нет доступной динамической памяти достаточного размера;

SS$_INVCHAN - неверное содержимое длинного слова контекста;

SS$_IVIDENT - указанный идентификатор имеет неверный формат;

SS$_NOIOCHAN - в базе данных прав нет больше доступных цепочек контекста;

SS¤_NOSUCHID - заданное имя идентификатора отсутствует в базе данных прав; или просмотрена уже вся база данных прав, если значение аргумента ID равно минус 1;

SS¤_PRV - пользователь не имеет доступа с чтением в базу данных прав.

Поскольку база данных прав является индексным файлом, доступ к которому осуществляется с помощью системы управления данными СУД-32, данная программа служебного обслуживания может также возвращать коды состояния СУД-32 (см. документ [1]).

13.71. ¤LCKPAG - зафиксировать страницы в памяти

Программа системного обслуживания ¤LCKPAG фиксирует одну страницу или диапазон страниц в физической памяти. Указанные виртуальные страницы принудительно включаются в рабочий набор и затем фиксируются в памяти. Зафиксированная страница не выгружается из памяти, если выгружается рабочий набор ее процесса. Данные страницы не являются кандидатами на замещение страниц и в этом смысле они также зафиксированы в рабочем наборе.

Формат:

SYS$LCKPAG INADR,[RETADR],[ACMODE]

Аргументы:

INADR

Использование в МОС ВП: ADDRESS_RANGE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Начальный и конечный виртуальные адреса диапазона фиксируемых страниц. Аргумент INADR - это адрес массива из двух длинных слов, содержащего по порядку начальный и конечный виртуальные адреса процесса. В каждом виртуальном адресе используется только та часть, которая представляет номер виртуальной страницы; младшие 9 битов игнорируются.

Если начальный виртуальный адрес равен конечному, то фиксируется одна страница.

RETADR

Использование в МОС ВП: ADDRESS_RANGE

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Начальный и конечный виртуальные адреса страниц процесса, которые программа $LCKPAG фактически зафиксировала. Аргумент RETADR - это адрес массива из двух длинных слов, содержащего по порядку начальный и конечный виртуальные адреса процесса.

ACMODE

Использование в МОС ВП: ACCESS_MODE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Режим доступа, который требуется присвоить фиксируемым страницам. Аргумент ACMODE - это длинное слово, содержащее режим доступа. Макрокоманда $PSLDEF определяет четыре режима доступа.

Наиболее привилегированный режим доступа, который может использоваться - это режим доступа вызывающего процесса. Для успешного завершения программы $LCKPAG необходимо, чтобы результирующий режим имел равную или высшую привилегию по сравнению с режимом доступа, который уже присвоен фиксируемым страницам.

Описание

Для фиксации страниц в памяти вызывающий процесс должен иметь привилегию PSWAPM.

Если требуется зафиксировать несколько страниц и требуется конкретно определить, какие страницы были ранее захвачены, то следует фиксировать страницы по одной.

Если во время фиксации страниц возникла ошибка, то массив возвращаемых адресов, если он запрошен, показывает страницы, которые были успешно зафиксированы до возникновения ошибки. Если не было зафиксировано ни одной страницы, то оба длинных слова в массиве возвращаемых адресов содержат значение минус 1.

Страницы, зафиксированные в памяти, можно освободить при помощи программы системного обслуживания "освободить страницы в памяти" (=ULKPAG). При осуществлении выхода образа зафиксированные страницы автоматически освобождаются.

Возвращаемые значения кода состояния:

SS=_WASCLR - успешное завершение. Все указанные страницы были прежде свободными;

SS=_WASSET - успешное завершение. По крайней мере одна из указанных страниц была зафиксирована ранее;

SS=_ACCVIO - вызывающая программа не может прочитать массив входных адресов; либо какая-то страница в заданном диапазоне недоступна или не существует;

SS=_LCKPAGFUL - достигнут определенный в системе максимальный предел количества страниц, которые можно зафиксировать в памяти;

SS=_NOPRIV - процесс не имеет привилегии для фиксации страниц в памяти.

13.72. =LKWSET - фиксировать страницы в рабочем наборе

Программа системного обслуживания =LKWSET фиксирует диапазон страниц в рабочем наборе; если данные страницы еще не включены в рабочий набор, то она включает и фиксирует их. Страница, которая зафиксирована в рабочем наборе, не

становится кандидатом на замещение.

Формат:

SYS$LKWSET INADR,[RETADR],[ACMODE]

Аргументы:

INADR

Использование в МОС ВП: ADDRESS_RANGE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Начальный и конечный виртуальные адреса диапазона фиксируемых в рабочем наборе страниц. Аргумент INADR - это адрес массива из двух длинных слов, содержащих по порядку начальный и конечный виртуальные адреса процесса. В каждом виртуальном адресе используется только та часть, которая представляет номер виртуальной страницы. Младшие 9 битов игнорируются.

Если начальный виртуальный адрес равен конечному, то фиксируется одна страница.

RETADR

Использование в МОС ВП: ADDRESS_RANGE

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Начальный и конечный виртуальный адреса диапазона страниц, фактически зафиксованных программой $LVKWSET. Аргумент RETADR - это адрес массива из двух слов, содержащих по порядку начальный и конечный виртуальные адреса про-

цесса.

ACMODE

Использование в МОС ВП: ACCESS_MODE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Режим доступа, который требуется присвоить фиксируемым страницам. Аргумент ACMODE - это длинное слово, содержащее режим доступа. Макрокоманда $PSLDKF определяет четыре режима доступа.

Наиболее привилегированный режим доступа, который может использоваться - это режим доступа вызывающего процесса. Для успешного завершения программы системного обслуживания $LKWSET необходимо, чтобы результирующий режим доступа имел равную или высшую привилегию по сравнению с тем режимом доступа, который уже присвоен фиксируемым страницам.

Описание

Если требуется зафиксировать несколько страниц и требуется конкретно определить, какие страницы уже были ранее зафиксированы, то следует фиксировать страницы по одной.

Если во время фиксации страниц возникла ошибка, то массив возвращаемых адресов, если он запрошен, показывает страницы, которые были успешно зафиксированы до возникновения ошибки. Если не было зафиксировано ни одной страницы, то оба длинных слова в массиве возвращаемых адресов страниц содержат минус 1.

Страницы, зафиксированные в рабочем наборе, можно освободить при помощи программы системного обслуживания "освободить страницы в рабочем наборе" (¤ULWSET).

Глобальные страницы с доступом на запись нельзя фиксировать в рабочем наборе.

Возвращаемые значения кода состояния:

SS¤_WASCLR - успешное завершение. Все заданные страницы были прежде свободны;

SS¤_WASSET - успешное завершение. По крайней мере одна из указанных страниц была ранее зафиксирована в рабочем наборе;

SS¤_ACCVIO - вызывающая программа не может прочитать массив входных адресов или не может записать массив выходных адресов; либо какая-то страница в заданном диапазоне недоступна или не существует;

SS¤_LKWSETFUL - рабочий набор полон. Если фиксируется несколько страниц, то, возможно, не хватает доступных динамических страниц для того, чтобы продолжить выполнение;

SS¤_NOPRIV - какая-то страница в заданном диапазоне принадлежит системному адресному пространству, либо указана глобальная страница с доступом на запись;

SS¤_PAGOWNVIO - страницы нельзя зафиксировать, потому что режим доступа, связанный с вызовом ¤LKWSET, является менее привилегирован-

ным по сравнению с тем режимом доступа, который связан с фиксируемыми страницами.

13.73. ¤MGBLSC - отобразить глобальную секцию

Программа системного обслуживания ¤MGBLSC устанавливает соответствие (отображает) между страницами в виртуальном адресном пространстве процесса и физическими страницами, занятыми под глобальную секцию.

Формат:

SYS¤MGBLSC INADR,[RETADR],[ACMODE],[FLAGS],
GSDNAME,[IDENT],[RELPAG]

Аргументы:

INADR

Использование в МОС ВП: ADDRESS_RANGE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Начальный и конечный виртуальные адреса в виртуальном адресном пространстве процесса (либо в области P0, либо в области P1), на которые необходимо отобразить секцию. Аргумент INADR - это адрес массива из двух длинных слов, содержащих по порядку начальный и конечный виртуальные адреса процесса. В каждом виртуальном адресе используется только та часть, которая представляет номер виртуальной страницы; младшие 9 битов игнорируются.

Если начальный виртуальный адрес равен конечному, то отображается одна страница (за исключением того случая, когда в аргументе FLAGS установлен бит SEC$M_EXPREG).

Если в аргументе FLAGS установлен бит SEC$M_EXPREG, то начальный адрес (первое длинное слово), заданный в аргументе INADR, определяет, отображается ли данная секция в область программы (P0) или в область управления (P1); конечный адрес (второе длинное слово) игнорируется.

RETADR

Использование в МОС ВП: ADDRESS_RANGE

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Начальный и конечный виртуальные адреса процесса, на которые $MGBLSC фактически отобразила секцию. Аргумент RETADR - это адрес массива из двух длинных слов, содержащих по порядку начальный и конечный виртуальные адреса процесса.

ACMODE

Использование в МОС ВП: ACCESS_MODE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Режим доступа, который требуется присвоить страницам, отображаемым в виртуальное адресное пространство процесса. Аргумент ACMODE - это длинное слово, содержащее режим доступа. Макрокоманда $PSLDEF определяет обозначения для четы-

рех режимов доступа

FLAGS

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Маска флагов, задающая варианты операции. Аргумент FLAGS - это битовый вектор размером в длинное слово, где каждый установленный в 1 бит задает соответствующую возможность.

Символические имена для битов определяют при помощи макрокоманды $SECDEF. Чтобы построить аргумент FLAGS, необходимо указать символические имена каждой желаемой возможности в операции "логическое или".

В следующем списке описаны значения флагов:

1) SEC$M_WRT - отобразить секцию с доступом на чтение/запись. По умолчанию секция отображается с доступом только на чтение;

2) SEC$M_SYSGBL - отобразить системную глобальную секцию. По умолчанию секция является групповой глобальной секцией;

3) SEC$M_EXPREG - отобразить секцию на первый доступный диапазон виртуальных адресов. По умолчанию секция отображается на диапазон, заданный аргументом INADR.

GSDNAM

Использование в МОС ВП: SECTION_NAME

Тип: строка текста

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Аргумент GSDNAM - это адрес дескриптора символьной строки, указывающего на строку имени глобальной секции.

Для групповых глобальных секций операционная система МОС ВП интерпретирует UIC группы как часть имени глобальной секции; таким образом, все имена глобальных секций неявно квалифицируются их полями идентификации.

IDENT

Использование в МОС ВП: SECTION_ID

Тип: квадрослово (без знака)

Доступ: только чтение

Механизм: по ссылке

Значение идентификации, задающее номер версии для процессов, отображающих существующую глобальную секцию, критерий соответствия идентификаций. Аргумент IDENT - это адрес структуры размером в квадрослово, содержащей три поля.

Первое длинное слово задает в трех младших битах критерий соответствия. Допустимые значения критерия, символические имена, при помощи которых их можно задавать, и их значения, перечислены в табл.64.

Таблица 64

Значение	Имя	Критерий соответствия
0	SEC$K_MATALL	соответствуют все версии секции
1	SEC$K_MATEQU	соответствуют только те секции, для которых совпадает младшая и старшая идентификации
2	SEC$K_MATLEQ	соответствуют только те секции, для которых старшие идентификации равны, а младшая идентификация, заданная отображающим процессом, меньше или равна младшей идентификации глобальной секции

Во втором длинном слове содержится номер версии. Номер версии содержит два поля: младшую идентификацию в младших 24-х битах и старшую идентификацию в старших 8-ми битах.

Если аргумент IDENT не задан или равен 0, (по умолчанию), то поля номера версии и критерия соответствия по умолчанию принимаются равными 0.

RELPAG

Использование в МОС ВП: LONGWORD_UNSIGNED

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Относительный номер первой отображаемой страницы в секции. Аргумент RELPAG - это длинное слово, содержащее

данный номер.

Если аргумент RELPAG не задан или равен 0 (по умолчанию), то глобальная секция отображается, начиная с первого виртуального блока в секции.

Описание

Маска защиты, задаваемая во время создания глобальной секции, определяет тип доступа (например, чтение/запись или только чтение), который имеет к секции отдельный процесс.

Программа ¤MGBLSC использует следующие системные ресурсы:

1) квота предела рабочего набора процесса (WSQUOTA) должна быть достаточной для увеличивающегося при отображении секции размера виртуального адресного пространства;

2) если страницы секции являются копируемыми по ссылке, то процесс должен также иметь достаточную квоту файла обмена страниц (PQFLQUOTA).

Использование данной программы системного обслуживания приводит к тому, что рабочий набор вызывающего процесса принимает размер, установленный квотой рабочего набора (WSQUOTA). Если размер рабочего набора процесса меньше данной квоты, то размер рабочего набора увеличивается. Если размер рабочего набора процесса больше данной квоты, то размер рабочего набора уменьшается.

Когда программа ¤MGBLSC отображает глобальную секцию, она добавляет страницы к виртуальному адресному пространству процесса. Секция отображается в направлении от младших адресов к старшим независимо от того, отображается данная

секция в зону программы или в зону управления.

Если во время отображения секции возникла ошибка, то массив возвращаемых адресов, если он задан, показывает страницы, которые были успешно отображены до возникновения ошибки. Если не было отображаемых страниц, то оба длинных слова в массиве возвращаемых адресов содержат минус 1.

Возвращаемые значения кода состояния:

SS$_NORMAL - успешное завершение;

SS$_ACCVIO - вызывающая программа не может прочитать массив входных адресов, имя глобальной секции или дескриптор имени, или поле идентификации секции, либо не может записать массив возвращаемых адресов;

SS$_ENDOFFILE - предупреждение. Заданный номер начального виртуального блока находится за пределами логического конца файла;

SS$_EXQUOTA - процесс превысил свою квоту файла обмена страниц, создавая копируемые по ссылке страницы;

SS$_INSFWSL - предел рабочего набора процесса мал для увеличившегося виртуального адресного пространства;

SS$_INTERLOCK - захват битовой маски для назначения глобальных секций из указанной разделяемой памяти осуществлен другим про-

цессом;

SS$_IVLOGNAM - имя глобальной секции имеет нулевую длину или длину более 15 символов;

SS$_IVSECFLG - установлен зарезервированный флаг;

SS$_IVSECIDCTL - недопустимое поле управления критерием в идентификации глобальной секции;

SS$_NOPRIV - маска защиты файла, задававшаяся при создании глобальной секциии, противоречит типу доступа, запрошенному вызывающим процессом; или какая-то страница из диапазона входных адресов находится в системном адресном пространстве;

SS$_NOSUCHSEC - предупреждение. Заданная глобальная секция не существует;

SS$_PAGOWNVIO - страница из указанного диапазона входных адресов принадлежит более привилегированному режиму доступа;

SS$_SHMNOTCNCT - разделяемая память, имя которой задано в аргументе GSDNAM, неизвестна системе. Данная ошибка может возникнуть в результате орфографической ошибки в строке, неверно присвоенного логического имени, или в результате сбоя при идентификации памяти, как разделяемой, во время системной гене-

рации;

SS¤_TOOMANYLNAM - трансляция логического имени для строки GSDNAM превысила допустимую глубину;

SS¤_VASFULL - переполнено виртуальное адресное пространство процесса; нет места в таблице страниц для страниц, которые создаются под отображаемую глобальную секцию.

13.74. ¤MOD_HOLDER - модифицировать запись держателя в базе данных прав

Программа служебного обслуживания ¤MOD_HOLDER модифицирует указанную запись держателя соответствующего идентификатора в базе данных прав. При этом можно добавлять или удалять атрибуты идентификатора.

Формат:

SYS¤MOD_HOLDER ID,HOLDER,[SET_ATTRIB],
[CLR_ATTRIB]

Аргументы:

ID

Использование в МОС ВП: RIGHTS_ID

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Двоичное значение того идентификатора, для которого программа ¤MOD_HOLDER модифицирует запись держателя. Аргу-

мент ID является длинным словом, содержащим значение идентификатора.

HOLDER

Использование в МОС ВП: RIGHTS_HOLDER

Тип: квадрослово (без знака)

Доступ: только чтение

Механизм: по ссылке

Идентификатор держателя, модифицируемый в результате выполнения программы системного обслуживания $MOD_HOLDER. Аргумент HOLDER является адресом квадрослова, содержащего идентификатор UIC держателя в первом длинном слове и нулевое значение во втором длинном слове.

SET_ATTRIB

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Битовая маска атрибутов, которые устанавливаются для идентификатора в результате работы программы $MOD_HOLDER. Аргумент SET_ATTRIB является длинным словом, содержащим маску атрибутов.

В системной макробиблиотеке (макрокоманда $KGBDEF) определено символическое значение:

KGB$V_RESOURCE - позволяет держателю запрашивать для идентификатора ресурсы типа блоков на дисках.

Символическое значение является смещением в битах внутри длинного слова. Данное значение можно также получить

в виде маски с соответствующим установленным битом, если вместо префикса KGB$V использовать префикс KGB$M.

CLR_ATTRIB

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Битовая маска атрибутов, которые должны быть отключены для идентификатора в результате работы программы $MOD_HOLDER. Аргумент CLR_ATTRIB является длинным словом, содержащим данную маску атрибутов.

В системной макробиблиотеке (макрокоманда $KGBDEF) определяется символическое значение:

1KGB$V_RESOURCE - позволяет держателю запрашивать для идентификатора ресурсы типа блоков на дисках.

Символическое значение является смещением в битах внутри длинного слова. Данное значение можно также получить в виде маски с соответствующим установленным битом, если вместо префикса KGB$V использовать префикс KGB$M.

Описание:

Программа системного обслуживания "модифицировать запись держателя в базе данных прав" модифицирует указанную запись держателя в базе данных прав. При этом можно добавлять или удалять атрибуты идентификатора.

Если заданы оба аргумента SET_ATTRIB и CLR_ATTRIB, то сначала бит сбрасывается. Таким образом, если задать один и тот же вид атрибута с каждым аргументом, то результатом

будет установка данного бита.

Для использования данной программы системного обслуживания требуется доступ с записью в базу данных прав. Если данная база данных находится в группе файлов SYS¤SYSTEM (что принимается по умолчанию), то для получения доступа с записью в такую базу данных требуется привилегия SYSPRV.

Возвращаемые значения кодов состояния:

SS¤_NORMAL - успешное завершение;

SS¤_ACCVIO - вызывающая программа не может прочитать аргумент HOLDER;

SS¤_BADPARAM - указанные атрибуты содержат неверные флаги атрибутов;

SS¤_INSFMEM - для открытия базы данных прав нет доступной динамической памяти достаточного размера;

SS¤_IVIDENT - заданный идентификатор или идентификатор держателя имеет неверный формат;

SS¤_NOSUCHID - указанный идентификатор отсутствует в базе данных прав; или указанный идентификатор держателя отсутствует в базе данных прав;

RMS_PRV - пользователь не имеет доступа с записью в базу данных прав.

Поскольку база данных прав является индексным файлом, доступ к которому осуществляется с помощью системы управления данными СУД-32, данная программа системного обслуживания может также возвращать коды состояния системы СУД-32,

связанные с операциями над индексными файлами (см. документ [1]).

13.75. ¤MOD_IDENT модифицировать идентификатор в базе данных прав

Программа системного обслуживания ¤MOD_IDENT модифицирует указанную запись идентификатора в базе данных прав. При этом можно добавить или удалить атрибуты идентификатора, изменить имя или значение идентификатора.

Формат:

SYS¤MOD_IDENT ID,[SET_ATTRIB],[CLR_ATTRIB]
,[NEW_NAME],[NEW_VALUE]

Аргументы:

ID

Использование в МОС ВП: RIGHTS_ID

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Двоичное значение идентификатора, запись которого модифицируется в результате работы программы ¤MOD_IDENT. Аргумент ID является длинным словом, содержащим значение идентификатора.

SET_ATTRIB

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Битовая маска атрибутов, которые включаются для идентификатора в результате работы программы ¤MOD_IDENT. Аргумент SET_ATTRIB является длинным словом, содержащим маску атрибутов.

В системной библиотеке (макрокоманда ¤KGBDEF) определяется символическое значение:

KGB¤V_RESOURCE - позволяет держателю запрашивать для идентификатора ресурсы типа блоков на дисках.

Символическое значение является смещением в битах внутри длинного слова. Данное значение можно также получить в виде маски с соответствующим установленным битом, если вместо префикса KGB¤V использовать префикс KGB¤M.

CLR_ATTRIB

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Битовая маска атрибутов, которые должны быть выключены для идентификатора в результате работы программы ¤MOD_IDENT. Аргумент CLR_ATTRIB является длинным словом, содержащим маску атрибутов.

В системной макробиблиотеке (макрокоманда ¤KGBDEF) определяется символическое значение:

KGB¤V_RESOURCE - позволяет держателю запрашивать для идентификатора ресурсы типа блоков на дисках.

Символическое значение является смещением в битах внутри длинного слова. Данное значение можно получить также

в виде маски с соответствующим установленным битом, если вместо префикса KGB$V использовать префикс KGB$M.

NEW_NAME

Использование в МОС ВП: CHAR_STRING

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Новое имя, которое должно быть назначено указанному идентификатору. Аргумент NEW_NAME является адресом дескриптора, указывающего на строку имени идентификатора.

Имя идентификатора содержит от 1 до 31 символа, включая знаки денежной единицы и подчеркивания, при этом в имя обязательно должен входить по крайней мере один нецифровой символ. Если заданы какие-то строчные буквы, то все они автоматически преобразовываются в прописные буквы.

NEW_VALUE

Использование в МОС ВП: RIGHTS_ID

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Новое значение, которое должно быть назначено указанному идентификатору. Аргумент NEW_VALUE является длинным словом, содержащим двоичное значение указанного идентификатора. При изменении значения идентификатора программа $MOD_IDENT меняет также значение данного идентификатора во всех записях держателей, в которых имеется ссылка на данный

идентификатор.

Описание

Программа системного обслуживания "модифицировать идентификатор в базе данных прав" модифицирует указанную запись идентификатора в базе данных прав. Если заданы оба аргумента SET_ATTRIB и CLR_ATTRIB, то атрибут сначала удаляется. Таким образом, если один и тот же атрибут указан в аргументах SET_ATTRIB и CLR_ATTRIB, то в результате этого бит будет установлен.

Для использования данной программы системного обслуживания требуется доступ с записью в базу данных прав. Если база данных входит в группу файлов SYS¤SYSTEM (что принимается по умолчанию), то для получения доступа с записью в базу данных требуется привилегия SYSPRV.

Возвращаемые значения кодов состояния:

SS¤_NORMAL - программа системного обслуживания успешно завершилась;

SS¤_NOSUCHID - указанный идентификатор отсутствует в базе данных прав;

SS¤_BADPARAM - указанные атрибуты содержат неверные флаги атрибутов;

SS¤_INSFMEM - для открытия базы данных прав нет доступной динамической памяти достаточного размера;

SS¤_IVIDENT - заданный идентификатор имеет неверный формат;

RMS¤_PRV - пользователь не имеет доступа с

записью в базу данных прав.

Поскольку база данных прав является индексным файлом, доступ к которому осуществляется с помощью системы управления данными СУД-32, данная программа системного обслуживания может также возвращать коды состояния системы СУД-32, связанные с операциями над индексными файлами (см. документ [1]).

13.76. ¤MOUNT - монтировать том

Программа системного обслуживания ¤MOUNT выполняет монтирование магнитной ленты, дискового тома или группы томов, а также задает варианты операции монтирования.

Формат:

SYS¤MOUNT ITMLST

Аргументы:

ITMLST

Использование в МОС ВП: ITEM_LIST_3

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Список элементов, задающий параметры для программы MOUNT. Аргумент ITMLST является адресом списка дескрипторов элементов, каждый из которых предоставляет информацию, необходимую для выполнения операции монтирования.

Список элементов должен включать в себя по крайней мере один дескриптор устройства. Список элементов завершается длинным словом, содержащим значение 0.

На рис.70 представлен формат дескриптора элемента.

Дескриптор элемента

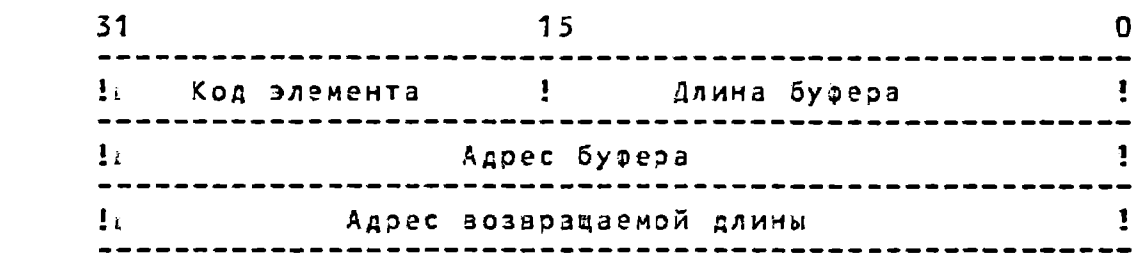


Рис. 70

Поля дескриптора элемента программы ¤MOUNT:

длина буфера -

Слово, задающее длину (в байтах) буфера, который предоставляет информацию, необходимую программе ¤MOUNT, чтобы обработать данный код элемента. Требуемая длина буфера зависит от кода элемента. Если значение поля "длина буфера" слишком мало, то программа ¤MOUNT усекает данные;

код элемента -

Слово, содержащее предоставляемый пользователем символический код, который задает информацию для операции монтирования. Данные коды определяются макрокомандой ¤MNTDEF;

адрес буфера -

Длинное слово, содержащее адрес буфера, который предоставляет информацию программе ¤MOUNT;

адрес возвращаемой длины -

Данное поле не используется;

Коды элементов программы MOUNT:

MNT¤_DEVNAM -

Данный код элемента задает имя устройства, которое должно монтироваться. Буфер должен содержать строку символов длиной от 1 до 64 байтов, представляющую имя устройства. Имя устройства может быть именем физического устройства или логическим именем. Если это логическое имя, то оно должно транслироваться в имя физического устройства. Если монтируется группа томов, то данный код появляется в списке более одного раза, поскольку в данном случае для каждого тома из группы томов монтируется одно устройство;

MNT¤_VOLNAM -

Данный код элемента задает имя тома, который должен быть смонтирован на устройстве. Буфер должен содержать строку длиной от 1 до 12 символов, которая является именем тома. Код элемента MNT¤_VOLNAM может появляться в списке более одного раза, если монтируется группа томов, поскольку в данном случае каждому тому из группы томов дается одно имя тома. Если монтируется группа томов, то для каждого тома из группы томов должны быть указаны коды MNT¤_DEVNAM и MNT¤_VOLNAM. Программа MOUNT монтирует том, указанный в первом коде элемента MNT¤_VOLNAM, на устройстве, указанном в первом коде элемента MNT¤_DEVNAM списка; том, указанный во втором коде элемента MNT¤_VOLNAM, монтирует на устройстве, указанном во втором коде элемента MNT¤_DEVNAM и и.д.

Для всех заданных томов и устройств. Таким образом в списке должно быть равное число данных двух кодов элементов. Если монтируется группа томов магнитных лент, то число кодов элементов MNT¤_DEVNAM не обязательно должно быть равно числу кодов элемента MNT¤_VOLNAM, т.к. На одном устройстве можно монтировать несколько таких томов;

MNT¤_LOGNAM -

Данный код элемента задает логическое имя тома. Данное логическое имя становится эквивалентным имени устройства, заданному первым кодом MNT¤_DEVNAM. Буфер должен содержать строку длиной от 1 до 64 символов, которая является логическим именем. Если не задан вариант (MNT¤M_GROUP или MNT¤M_SYSTEM), то логическое имя вводится в таблицу логических имен процесса;

MNT¤_FLAGS -

Данный код элемента задает вектор размером в длинное слово, в котором каждый бит указывает вариант операции монтирования. Буфер должен содержать длинное слово, которое является вектором битов.

Макрокоманда ¤MNTDEF определяет символические имена для каждого варианта в векторе битов. Вектор битов конструируется путем указания символических имен для нужных вариантов в операции "логическое или".

Символические имена для всех битов следующие:

MNT$M_FOREIGN - том должен монтироваться как чужой; это означает, что том не имеет файловой структуры. Если указан данный вариант, то следующие коды элементов могут появляться в списке только один раз: MNT$_DEVNAM, MNT$_VOLNAM, MNT$_LOGNAM. Чтобы задать данный вариант, вызывающий процесс должен либо быть владельцем данного тома, либо иметь привилегию VOLPRO;

MNT$M_GROUP - логическое имя тома, который должен монтироваться, вводится в таблицу логических имен группы, и том делается доступным другим пользователям с тем же номером группы UIC, что и вызывающий процесс. Чтобы задать данный вариант, пользователь должен иметь привилегию GRPNAM. Данный вариант относится только к дискам;

MNT$M_NOASIST - программа $MOUNT не запрашивает вмешательства оператора, если во время операции монтирования произойдет ошибка. Если данный вариант не указан, то в некоторых ошибочных ситуациях программа $MOUNT запрашивает помощь у оператора для устранения данных ошибок;

MNT$M_NODISKQ - монтируемому тому не назязываются дисковые квоты. Если данный вариант не указан, то дисковые квоты назязываются. Чтобы задать данный вариант, вызывающий процесс должен быть либо владельцем тома, либо иметь привилегию VOLPRO. Данный вариант относится только к дискам;

MNT$M_NOHDR3 - при монтировании магнитных лент на них не должны записываться метки HDR3 и HDR4. Если данный вариант не указан, то соответствующие метки записываются на

все ленты;

MNT$M_NOWRITE - монтируемый том программно защищен от записи. Если данный вариант не указан, то предполагается, что том имеет доступ с записью с чтением;

MNT$M_OVR_ACCESS - если позволяет установка, то данный вариант замещает любой символ в поле доступности тома. Чтобы задать данный вариант, пользователь должен либо быть владельцем тома, либо иметь привилегию VOLPRO. Данный вариант относится только к магнитным лентам;

MNT$M_OVR_EXP - магнитная лента, для которой еще не истек срок хранения, может быть использована для записи другой информации. Чтобы указать данный вариант, пользователь должен быть либо владельцем данного тома, либо иметь привилегию VOLPRO;

MNT$M_OVR_IDENT - том можно монтировать, не указывая имя тома (используя код элемента MNT$_VOLNAM). Если данный вариант задан, то нельзя указывать следующие варианты: MNTM_GROUP, MNTM_SHARE и MNT$M_SYSTEM;

MNT$M_OVR_LOCK - программная блокировка записи, которая произошла, когда том имел испорченную битовую маску памяти, может быть изменена;

MNT$M_OVR_SETID - не должен выполняться контроль идентификации для группы томов, когда последовательно монтируются катушки с лентами из группы томов. Данный вариант относится только к магнитным лентам;

MNT$M_READCHECK - после каждой операции чтения должен выполняться контроль чтения;

MNT¤M_SHARE - том должен монтироваться как общий и, следовательно, том должен быть доступным другим пользователям. Данный вариант относится только к дискам. Если том был ранее смонтирован как общий другим пользователем, а в текущем вызове программы ¤MOUNT указывается вариант MNT¤_SHARE, то все остальные варианты, указанные в данном вызове, игнорируются. Если пользователь распределил данное устройство и указал при вызове программы ¤MOUNT вариант MNT¤_SHARE, то данная программа перераспределит устройство так, чтобы к тому имели доступ другие пользователи;

MNT¤M_MESSAGE - сообщения будут посылаться на устройство пользователя SYS¤OUTPUT;

MNT¤M_SYSTEM - логическое имя монтируемого тома вводится в системную таблицу логических имен и том становится доступным всем остальным пользователям при условии, что защита на базе идентификаторов UIC разрешает доступ к данному тому. Чтобы задать данный вариант, пользователь должен иметь привилегию SYSNAM. Данный вариант относится только к дискам.

MNT¤M_WRITECHECK - после каждой операции записи должен быть выполнен контроль записи;

MNT¤M_WRITETHRU - кэширование обратной записи отменяется, так что головные метки файлов записываются обратно на диск при каждой операции записи. Если данный вариант не задан, то головные метки файла кэшируются до тех пор, пока файл не закроется. Кэширование головных меток файла улучшает производительность при риске потери записанных данных

в случае сбоя системы. Данный вариант относится только к дискам;

MNT$M_NOMNTVER - том не помечается как кандидат для автоматической верификации монтирования. Если данный вариант не указан, то том помечается как кандидат для автоматической проверки монтирования. Данный вариант относится только к дискам;

MNT$M_NOCACHE - отменяется все кэширование, связанное с данным томом. Указание данного варианта эквивалентно заданию MNT$M_WRITETHRU, заданию величины 1 в дескрипторе элемента MNT$_FIELD и заданию величины 0 в дескрипторах элементов MNT$_EXTENT данный вариант относится только к дискам;

MNT$M_NOAUTO - отменяется автоматическая разметка томов (AVL) и автоматическое распознавание томов (AVR) если указан данный вариант, то оператор должен вводить команды с консоли для обработки каждого следующего тома из группы томов. После завершения обработки тома оператор указывает лентопротяжное устройство, на которое загружается следующий том, и имя метки следующего тома. Удобно использовать данный вариант для отключения средств AVL и AVR, если группа томов читается не последовательно. Пользователь может включить средства AVL и AVR указав вариант MNT$M_INIT_CONT. Вариант MNT$M_NOAUTO относится только к магнитным лентам;

MNT$M_INIT_CONT - последующие тома в группе томов должны инициализироваться без вмешательства оператора. Программа $MOUNT инициализирует новые тома с защитой,

заданной для первой магнитной ленты из группы томов и создает уникальные имена меток томов максимум для 99 томов из группы томов. Если задан данный вариант, то пользователь должен распределить ряд лентопротяжных устройств для группы магнитных томов. Если программа переключается на устройство, на которое не загружена магнитная лента или загружена не та магнитная лента, то она запрашивает у оператора загрузить правильную магнитную ленту. Программма ¤MOUNT демонтирует и разгружает тома после того как они обработаны. Оператор может загрузить следующий том из группы томов до того, как достигнет конца текущая катушка с магнитной лентной из данной группы томов. При записи на группу томов программа ¤MOUNT автоматически выполняет следующие операции: переключается на следующее устройство; инициализирует следующую магнитную ленту с теми же самыми меткой тома и защитой, как и первый том в той же группе; сообщает оператору о переключении. При чтении группы томов программа ¤MOUNT генерирует метку следующего тома и печатает данный том. Имя метки, которое программа ¤MOUNT генерирует для каждого последующего тома из группы томов, состоит из 6 символов: первые четыре символа совпадают с первыми четырьмя символами имени метки предыдущего тома в группе томов. Данный вариант относится только к магнитным лентам;

MNT¤M_OVR_VOLO- в метке тома не обрабатывать поле идентификатора владельца. Программа ¤MOUNT читает информацию о владельце тома из поля "владелец тома" меток тома. Операционная система МОС ВП требует, чтобы пользователь

указывал данный вариант для обработки магнитных лент когда том был создан на любой из операционных систем, отличных от МЭС ЭЛ или при инициализации тома была указана защита тома. Чтобы задать данный вариант вызывающий процесс должен иметь либо привилегию VOLPRO, либо быть владельцем данный тома. Данный вариант относится только к магнитным лентам;

MNT¤_ACCESSED -

Данный код элемента задает число одновременно используемых каталогов на томе. Буфер должен содержать целое число в длинном слове в диапазоне от 0 до 255. Данное число замещает то число каталогов, которое было указано при инициализации тома. Чтобы указать код элемента MNT¤_ACCESSED, взывающий процесс должен иметь привилегию OPER. Данный вариант относится только к дискам;

MNT¤_PROCESSOR -

Для магнитных лент и дисков с файловой структурой уровня 1 код элемента задает имя вспомогательного управляющего процесса (ACP), который должен обрабатывать данный том. Указанный процесс ACP замещает тот, который связан с данным устройством. Для дисков с файловой структурой уровня 2 данный элементный код элемента управляет распределением кэша блоков. Чтобы задать код элемента MNT¤_PRCESSOR, вызывающий процесс должен иметь привилегию OPER. Буфер дожен содержать строку символов, задающую либо строку UNIQUE, либо имя устройства, либо спецификацию файла.

Действия, которые предпринимаются в каждом из данных случаев следующие:

1) UNIQUE - для дисков с файловой структурой уровня 1 строка UNIQUE указывает, что программа ¤MOUNT создает новый процесс для выполнения копии образа ACP, принятого по умолчанию, который связан с устройством, заданным элементным кодом MNT¤_DEVNAM. Для дисков с файловой структурой уровня 2 строка UNIQUE распределяет отдельный кэш блоков;

2) DDCU: - для магнитных лент и дисков с фэйловой структурой уровня 1 строка DDCU: указывает, что программа ¤MOUNT использует тот процесс ACP, который используется в текущий момент устройством DDCU. Заданное устройство должно иметь формат DDCU:, например, DUA3:. Для дисков с файловой структурой уровня 2 строка указывает, что программа ¤MOUNT берет распределение блоков из заданного устройства;

3) FILE_SPEC - указывает, что программа ¤MOUNT создает новый процесс для выполнения образа ACP со спецификацией фэйла FILE_SPEC. Символы обобщения недопустимы. Файл должен быть расположен на диске и каталоге, заданном логическим именем SYS¤SYSTEM;

MNT¤_VOLSET -

Данный код элемента задает имя группы томов. Буфер должен содержать строку, содержащую от 1 до 12 алфавитно-цифровых символов, которые составляют имя группы томов. Если указан данный код элемента, то тома, заданные кодом MNT¤_VOLNAM, связываются в новую группу томов или добавляются к уже существующей группе

в зависимости от того, является ли указанное имя в коде эдемента MNT¤_VOLSET, новым или уже существующим именем. Если код элемента MNT¤_VOLSET задается для добавления томов к существующей группе томов, то корневой том (RVN1) должен либо быть уже смонтированным, либо указанным первым в списке (кодами MNT¤DEVNAM и MNT¤_VOLNAM). Если код элемента MNT¤_VOLSET задается для создания новой группы томов, то первый том, заданный в списке кодами элементов MNT¤_DEVNAM и MNT¤_VOLNAM, становится корневым томом;

MNT¤_BLOCKSIZE -

Данный код элемента задает размер блока, принимаемый по умолчанию для томов на магнитных лентах. Буфер должен содержать целое число размером в длинное слово в диапазоне от 20 до 65532 байтов для работы с системой управления данными СУД-32, или от 10 до 65534 байтов для операций, которые не используют систему СУД-32. Данный код элемента относится только к магнитным лентам. Если данный код элемента не задан, то по умолчанию принимается длина блока 2048 байтов для томов на магнитных лентах файловой структуры и 512 байтов для "чужих" и непомеченных магнитных лент. Код элемента MNT¤_BLOCKSIZE нужно указывать при монтировании магнитных лент, которые не имеют меток HDR2 и магнитных лент, которые должны содержать записи, которые больше длины блока, принимаемой по умолчанию;

MNT¤_DENSITY -

Данный код элемента задает плотность, с которой данные должны записываться на "чужой" или непомеченный том на магнитной ленте. Буфер должен содержать значение размером в длинное слово, которое задает одну из следующих допустимых плотностей: 800, 1600 или 6250. Данный код элемента относится только к магнитным лентам. Указанная плотность будет использоваться только, если лента является "чужой", непомеченной и первой операцией является запись;

MNT¤_EXTENT -

Данный код элемента задает размер кэша экстентов в единицах указателей экстентов. Буфер должен содержать длинное слово, значение которого задает данный размер. Чтобы указать данный код элемента требуется привилегия OPER. Значение 0 (принимаемое по умолчанию) отменяет кэширование. Данный код элемента относится только к дискам;

MNT¤_FILEID -

Данный код элемента задает размер кэша идентификаторов файлов в единицах номеров файлов. Размер должен содержать длинное слово, значение которого задает данный размер. Чтобы указать данный код элемента, требуется привилегия OPER. Значение 1 отменяет кэширование. Данный код элемента относится только к дискам;

MNT¤_LIMIT -

Данный код элемента задает максимальную величину сво-

бодного пространства в кэше экстентов. Буфер должен содержать длинное слово, значение которого задает размер свободного пространства на диске. Данный код элемента относится только к дискам;

MNT$_OWNER -

Данный код элемента задает идентификатор UIC, который должен быть назначен для владения томом. Буфер должен содержать в длинном слове восьмеричное значение, представляющее идентификатор UIC. Для назначения идентификатора UIC требуется либо привилегия VOLPRO, либо владение томом;

MNT$_VPROT -

Данный код элемента задает защиту, которая назначается тому. Буфер должен содержать в длинном слове маску защиты, которая определяет четыре типа доступа, разрешенные четырем категориям пользователей.

Маска защиты состоит из четырех 4-битовых полей. Каждое поле предоставляет или запрещает доступ с чтением (R), записью (W), логический (L) и физический (P) доступ для какой-то категории пользователей. Сброшенный бит разрешает доступ, установленный бит запрещает доступ. На рис.71 показана структура маски защиты.

Маска защиты

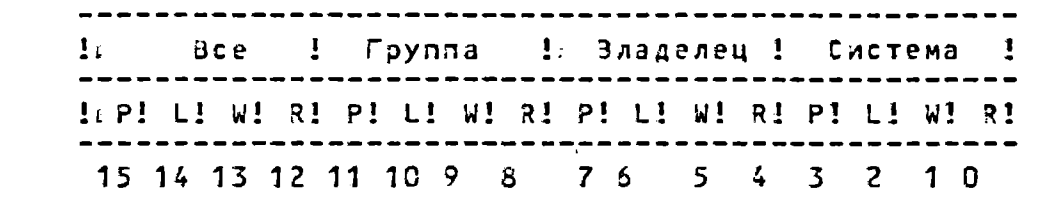


Рис. 71

Если код элемента MNT$_VPROT не задан или задан равным 0, то том получает защиту, которая ему была назначена при инициализации. Чтобы показать данный код элемента, вызывающий процесс должен либо быть владельцем данного тома, либо иметь привилегию VOLPRO;

MNT$_QUOTA -

Данный код элемента задает размер кэша записей квот в единицах записей квот. Буфер должен содержать длинное слово, значение которого задает данный размер. Для указания данного кода элемента требуется привилегия OPER. Значение 0 отменяет кэширование. Данный код элемента относится только к дискам;

MNT$_RECORDSIZ -

Данный код элемента задает количество символов в каждой записи. Данный код элемента используется вместе с кодом MNT$_BLOCKSIZE для указания форматов данных для "чужих" томов. Буфер должен содержать длинное слово, значение которого меньше или равно размеру блока. Код элемента MNT$_RECORDSIZ относится только к магнитным лентам. Если данный код не задан, то предполагается, что размер записи равен размеру блока;

MNT$_WINDOW -

Данный код элемента задает число указателей соответствия, которые должны быть распределены для окон файлов. Буфер должен содержать длинное слово, значение которого находится в диапазоне от 7 до 80. Данный значение замещает значение, принимаемое по умолчанию при инициализации тома. Данный код элемента относится только к дискам. Чтобы задать данный код, требуется привилегия OPER;

MNT$_EXTENTSION -

Данный код элемента задает число блоков, на которое могут быть расширены файлы. Буфер должен содержать длинное слово, значение которого находится в диапазоне от 0 до 65535. Данный код относится только к дискам;

MNT$_COMMENT -

Данный код задает текст, который должен быть связан с запросом к оператору. Буфер должен содержать строку символов длиной не более 78 символов. Данный текст будет печататься на консоли оператора при выдаче запроса к оператору для монтируемого устройства.

Описание:

Чтобы смонтировать конкретный том, пользователь, вызвавший программу $MOUNT, должен либо быть владельцем, либо иметь привилегию на доступ к указанному тому или томам. Требуемые привилегии зависят от операции и перечислены вместе с кодами элементов, которые задают конкретные операции.

Чтобы выполнить монтирование с помощью оператора, вызывающий процесс должен иметь привилегию TMPMBX или PRMMBX.

Программа ¤MOUNT для монтирования томов с групповым или системным доступом использует следующие ресурсы:

1) пул нестраничного обмена;

2) системный динамический пул.

Возвращаемые значения кода состояния:

SS¤_NORMAL - программа системного обслуживания успешно завершилась;

SS¤_ACCVIO - нет доступа к списку элементов или адресу, указанному в списке;

SS¤_BADPARAM - длина буфера равная нулю была указана вместе с ненулевым кодом элемента; или был указан недопустимый код элемента; или не было указано никакое устройство;

SS¤_NOGRPNAM - вызывающий процесс не имеет привилегии GRPNAM;

SS¤_NOSYSNAM - вызывающий процесс не имеет привилегии SYSNAM;

SS¤_NOOPER - вызывающий процесс не имеет требуемой привилегии OPER;

SS¤_NOPRIV - вызывающий процесс не имеет достаточной привилеги для доступа к указанному тому;

SS¤_NOSUCHDEV - предупреждение. Указанное устройство отсутствует.

Программа ¤MOUNT может также возвращать коды состояния, специфичные для средства монтирования. Данные значения кода состояния определяются макрокомандой символического определения ¤MOUNDEF (см. подраздел 2.3).

13.77. ¤MTACCESS - доступность магнитной ленты

Программа системного обслуживания ¤MTACCESS позволяет обеспечивать свою собственную программу для интерпретации и вывода поля доступности в метках VOL1 и HDR1 магнитных лент со стандартными метками.

Формат:

```
SYS¤MTACCESS     LBLNAM,[UIC],[STD_VERSION]
                 ,[ACCESS_CHAR],[ACCESS_SPEC]
                 ,TYPE
```

Аргументы:

LBLNAM

Использование в МОС ВП: ADDRESS

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Метка, которая должна обрабатываться. Аргумент LBLNAM является адресом длинного слова, содержащего данную метку. При вводе обрабатываемая метка является либо меткой VOL1, либо меткой HDR1, которые считываются с магнитной ленты. При выводе меток данное поле равно нулю. Тип обрабатываемой метки определяет аргумент TYPE.

UIC

Использование в МОС ВП: UIC

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Идентификатор UIC пользователя, выполняющего данную операцию. Аргумент UIC является длинным словом, содержащим данный UIC.

STD_VERSION

Использование в МОС ВП: LONGWORD_UNSIGNED

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Десятичный эквивалент версии стандарта, который считывается из метки VOL1. Аргумент STD_VERSION является длинным словом, содержащим номер версии стандарта.

ACCESS_CHAN

Использование в МОС ВП: LONGWORD_UNSIGNED

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Символ доступности, задаваемый пользователем. Аргумент ACCESS_CHAN представляет байт, содержащий символ доступности, используемый для вывода меток.

ACCESS_SPEC

Использование в МОС ВП: LONGWORD_UNSIGNED

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Значение, указывающее, был ли задан символ доступности, который передается в аргументе ACCESS_CHAN.

Аргумент ACCESS_SPEC представляет байт, содержащий одно из следующих значений:

1) MTA¤K_CHARVALID - да;

2) MTA¤K_NOCHAR - нет.

Данный аргумент используется только для вывода меток.

TYPE

Использование в МОС ВП: LONGWORD_UNSIGNED

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Тип поля доступности, которое должно обрабатываться. Аргумент TYPE представляет байт, содержащий одно из следующих значений:

1) MTA¤K_INVOL1 - ввод метки VOL1;

2) MTA¤K_INHDR1 - ввод метки HDR1;

3) MTA¤K_OUTVOL1 - ввод метки VOL1;

4) MTA¤K_OUTHDR1 - вывод метки HDR1.

Описание:

Программа системного обслуживания позволяет обеспечивать свою собственную программу для интерпретации и обработки поля доступности в метках VOL1 и HDR1 магнитных лент со стандартными метками. Можно заменить программу, принятую по умолчанию, установив флаг LOADMTACCESS программой SYSGEN. При перезагрузке операционной системы МОС ВП программа MTACCESS.EXE загружается в систему, а вектор диспетчера программ системного обслуживания устанавливается так, чтобы вызывать программу, предоставленную пользователем, вместо программы по умолчанию.

Программа, принятая по умолчанию, сначала проверяет в метке версию стандарта. Для магнитных лент с номером версии 3 или меньше программа выводит либо пробел, либо символ, заданный пользователем. При вводе с таких магнитных лент программа проверяет данное поле на пробел и возвращает код состояния SS$_FILACCERR, если значение поля отлично от пробела.

Для магнитных лент с номером версии больше 3, программа выводит либо символ, заданный аргументом ACCESS_CHAR, либо 1 в коде кои-8, если никакой символ не задан. При вводе с таких магнитных лент программа проверяет данное поле на пробел. Если поле содержит пробел, то регистр R0 устанавливается на 0. В данном случае пользователю дается полный доступ и защита операционной системы МОС ВП не работает. Если поле содержит 1 в коде кои-8, а поле VOL1 IMPLEMENTATION IDENTIFIER (идентификатор реализации) метки

VOL1 содержит системный код операционной системы МОС ВП, то в регистре R0 устанавливается код состояния SS¤_NORMAL. В данном случае контролируется защита операционной системы МОС ВП.

Если поле отлично от пробела и не содержит символ 1 в коде КОИ-8, то в регистре R0 устанавливается код состояния SS¤_FILACCERR, который побуждает пользователя заменить контроль доступности и позволяет системе управления файлами на магнитных лентах контролировать защиту операционной системы МОС ВП.

Возвращаемые значения кодов состояния:

SS¤_NORMAL - успешное завершение;

SS¤_FILACCERR - ошибка явной замены;

SS¤_NOFILACC - пользователь не имеет доступа к файлу;

SS¤_NOVOLACC - пользователь не имеет доступа к тому.

13.78. ¤NUMTIM - преобразовать двоичное время в цифровое

Программа системного обслуживания ¤NUMTIM преобразует абсолютное время или дельта-время из 64-битового системного формата времени в двоичные целые значения даты и времени.

Формат:

```
SYS¤NUMTIM      TIMBUF,[TIMADR]
```

Аргументы:

TIMBUF

Использование в МОС ВП: VECTOR_WORD_UNSIGNED

Тип: слово (без знака)

Доступ: только запись

Механизм: по ссылке

Буфер, в который $NUMTIM записывает преобразованные дату и время. Аргумент TIMBUF - это адрес структуры из 7 слов. Рис.72 показывает поля данной структуры.

Структура буфера TIMBUF

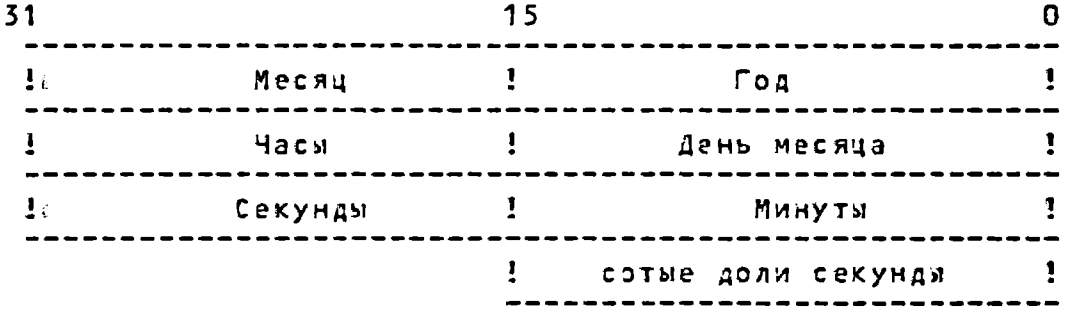


Рис. 72

Если аргумент TIMADR задает дельта-время, то $NUMTIM возвращает 0 в поля "Год" и "Месяц". В поле "День месяца" программа возвращает количество дней, заданных дельта-временем; данное количество должно быть меньше 10000 дней.

TIMADR

Использование в МОС ВП: DATE_TIME

Тип: квадрослово (без знака)

Доступ: только чтение

Механизм: по ссылке

64-битовое преобразуемое значение времени. Аргумент TIMADR - это адрес квадрослова, содержащего данное время. Положительное значение времени дает абсолютное время, отрицательное значение времени представляет дельта-время.

Если аргумент TIMADR не задан, то $NUMTIM возвращает текущее системное время.

Если аргумент TIMARD задан как 0, то $NUMTIM возвращает базовое время (17 ноября 1858 г.)

Возвращаемые значения кода состояния:

SS$_NORMAL - успешное завершение;

SS$_ACCVIO - вызывающая программа не может прочитать 64-битовое значение времени или не может записать в буфер;

SS$_IVTIME - заданное дельта-время равно или больше 10000 дней.

13.79. $PARSE_ALL - проанализировать элемент списка управления доступом

Программа системного обслуживания $PARSE_ALL анализирует заданную строку текста и преобразовывает ее в двоичное представление для элемента списка управления доступом (ACE).

Формат:

SYS$PARSE_ACL ACLSTR,ACLENT,[ERRPOS],[ACCNAM]

Аргументы:

ACLSTR

Использование в МОС ВП: CHAR_STRING

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Форматированный элемент ACE, анализ которого выполняется в результате работы программы. Аргумент ACLSTR является адресом дескриптора строки, указывающего на строку текста, предназначенную для анализа.

ACLENT

Использование в МОС ВП: CHAR_STRING

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Описание элемента ACE, который анализируется в результате работы данной программы. Аргумент ACLENT является адресом дескриптора, указывающего на буфер, в который записывается элемент ACE. Первый байт данного буфера содержит длину элемента ACE, второй байт содержит величину, которая идентифицирует тип элемента ACE, который в свою очередь определяет формат элемента ACE. Информация о типах элементов ACE и связанных с ними форматов приведена в описании программы системного обслуживания SYS$FORMAT_ACL.

ERRPOS

Использование в МОС ВП: WORD_UNSIGNED

Тип: слово (без знака)

Доступ: только запись

Механизм: по ссылке

Число символов из строки, определяемой аргументом ACLSTR, которые обработаны программой $PARSE_ACL. Аргумент ERRPOS является адресом слова, в которое записывается число символов, реально обработанных программой. Если программа заканчивается с ошибкой, то данное число указывает на то место в строке, в которой произошел сбой.

ACCNAM

Использование в МОС ВП: ACCESS_BIT_NAMES

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Имена битов в маске доступа при выполнении программы $PARSE_ACL. Аргумент ACCNAM является адресом массива из 32 квадрослов, которые определяют имена битов в маске доступа. Каждый элемент указывает на имя одного бита. Первый элемент дает имя биту 0, второй элемент дает имя биту 1 и т.д. Если аргумент ACCNAM опущен, то используются следующие имена:

Бит 0 - READ;

Бит 1 - WRITE;

Бит 2 - EXECUTE;

Бит 3 - DELETE;

Бит 4 - CONTROL;

Бит 5 - BIT_5;

Бит 6 - BIT_6;

Бит 31 - BIT_31.

Описание

Программа системного обслуживания "проанализировать элемент списка управления доступом" анализирует указанную строку текста и преобразовывает ее в двоичное представление для элемента управления доступом.

Возвращаемые значения кода состояния:

SS$_NORMAL - программа системного обслуживания успешно завершилась;

SS$_ACCVIO - невозможно прочитать строку или ее дескриптор; или невозможно прочитать дескриптор буфера; или невозможно записать в буфер; или буфер слишком мал, чтобы вместить элемент списка ACL;

SS$_IVACL - неверный формат списка управления доступом.

13.80. $PURGWS - очистить рабочий набор

Программа системного обслуживания $PURGWS удаляет заданный диапазон страниц из текущего набора вызывающего процесса для того, чтобы освободить место для страниц, которые требуются новому сегменту программы. Однако, прог-

рамма системного обслуживания "Изменить размер рабочего набора" (¤ADJWSL) представляет более предпочтительный механизм для управления использованием ресурсов физической памяти процессом.

Формат:

SYS¤PURGWS INADR

Аргументы:

INADR

Использование в МОС ВП: ADDRESS_RANGE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Начальный и конечный виртуальные адреса в диапазоне удаляемых страниц. Аргумент INADR - это адрес массива из двух длинных слов, содержащих по порядку начальный и конечный виртуальные адреса процесса. В каждом виртуальном адресе используется только та часть, которая представляет номер виртуальной страницы; младшие 9 битов игнорируются.

Программа системного обслуживания ¤PURGWS находит страницы в заданном диапазоне и удаляет их, если они находятся в рабочем наборе.

Если начальный виртуальный адрес равен конечному, то удаляется только одна страница.

Чтобы очистить рабочий набор целиком, необходимо задать диапазон страниц от 0 до 7FFFFFFF; в данном случае, образ будет продолжать выполняться и получать страничный отказ на требуемые ему страницы в рабочем наборе.

Возвращаемые значения кода состояния:

SS¤_NORMAL - успешное завершение;

SS¤_ACCVIO - вызывающая программа не может прочитать массив входных адресов.

13.31. ¤PUTMSG - вывести сообщение

Программа системного обслуживания ¤PUTMSG выводит одно или более сообщений об ошибке на устройство SYS¤ERROR (и на SYS¤OUTPUT, если оно отличается от SYS¤ERROR) программа ¤PUTMSG является обобщенной программой форматирования и вывода сообщений, которая используется операционной системой МОС ВП для записи информационных сообщений и сообщений об ошибках для процессов.

Формат:

SYS¤PUTMSG MSGVEC,[ACTRTN],[FACNAM],[ACTPRM]

Аргументы:

MSGVEC

Использование в МОС ВП: CNTRLBLK

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Вектор аргументов сообщения, который определяет выводимое сообщение (или сообщения), а также поля сообщения, которые должна использовать программа ¤PUTMSG при записи сообщения (или сообщений). Аргумент MSGVEC является адресом вектора сообщения.

Вектор сообщения состоит из одного длинного слова, за которым следует один или более дескрипторов сообщения, по одному дескриптору на сообщение. Рис.74 показывает содержимое первого длинного слова. 3

Вектор сообщения

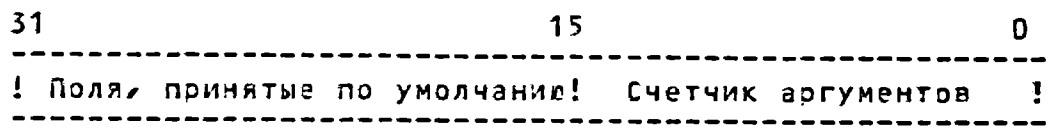


Рис. 73

Поля вектора сообщений:

счетчик аргументов -

Данное поле длиной в слово задает общее число длинных слов в векторе сообщения, не включая первого длинного слова (частью которого оно является);

поля, принятые по умолчанию -

Данное поле длиной в слово указывает, какие компоненты сообщения будут записываться. Поле "Поля сообщения, принятые по умолчанию" представляют из себя вектор битов размером в слово, в котором каждый установленный бит указывает, что должна выводиться соответствующая компонента сообщения.

В табл.65 приведены номера значащих битов. Приведенные номера битов (0, 1, 2 ,3) являются битовыми позициями от начала слова, однако поскольку данное слово является вторым в длинном слове, то, чтобы определить правильное смещение бита внутри длинного слова, следует прибавить к его номеру число 16.

Таблица 65

Бит	Значение	Описание
0	1	включить текст сообщения
	0	не включать текст сообщения
1	1	включить мнемоническое имя для текста сообщения
	0	не включать мнемоническое имя для текста сообщения
2	1	включить индикатор уровня серьезности
	0	не включать индикатор уровня серьезности
3	1	включить префикс средства
	0	не включать префикс средства

Биты с 4 по 15 должны быть равны 0.

Данную установку по умолчанию, заданную полем "Поля сообщения, принятые по умолчанию", можно изменить для любого или для всех сообщений, задав другие поля сообщений в поле "Новые поля сообщения" любого последующего дескриптора сообщения. Если задано поле "Новые поля сообщения", то указанные поля становятся новой установкой по умолчанию для всех остальных сообщений, пока снова не будет задано поле "Новые поля сообщения".

Программа ¤PUTMSG передает поле "Флаги сообщения, принятые по умолчанию" программе системного обслуживания ¤GETMSG в качестве аргумента FLAGS.

Если поле "Флаги сообщения, принятые по умолчанию" не задано, то используются поля сообщения по умолчанию для процесса. Поля сообщения по умолчанию для процесса могут быть установлены с помощью команды SET MESSAGE языка DCL.

Вслед за первым длинным словом вектора сообщения идут один или более дескрипторов сообщений. Дескриптор сообщения может иметь один из четырех возможных форматов, в зависимости от типа сообщения, которое он описывает. Существуют следующие типы сообщений:

1) сообщения, предоставляемые пользователем;

2) системные сообщения;

3) сообщения системы СУД-32;

4) сообщения об исключительных состояниях системы.

Дескриптор сообщения для сообщений, предоставленных пользователем, показан на рис.74.

Дескриптор сообщения для сообщений, предоставленных пользователю

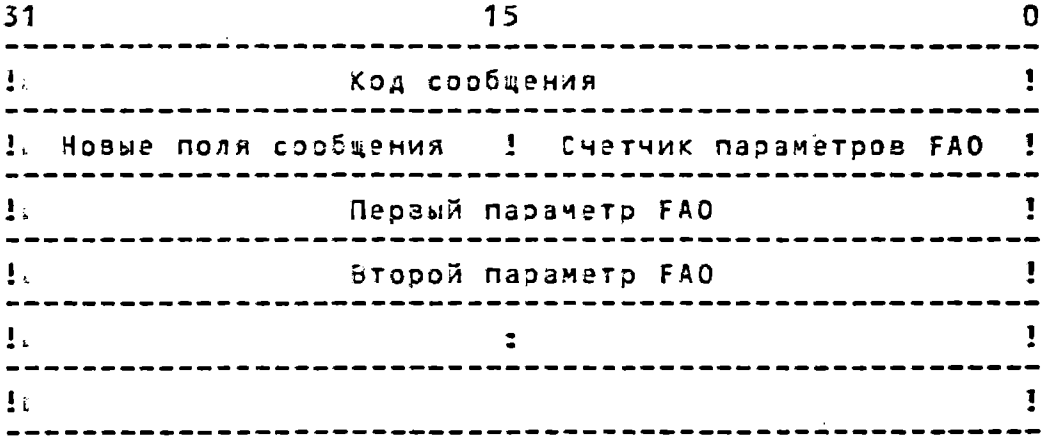


Рис. 74

Поля в дескрипторе сообщения для сообщений, предоставленных пользователем:

код сообщения -

Длинное слово, значение которого уникально идентифицирует сообщение;

счетчик параметров FAO -

Значение длинного слова, задающее число длинных слов с параметрами FAO, которые следуют за дескриптором сообщения. Число необходимых параметров FAO зависит от директив FAO, используемых в тексте сообщения. Некоторые из директив FAO требуют одного или более параметров, а другие директивы не требуют ни одного параметра;

новые поля сообщения -

Вектор битов длиной в слово, задающий новые поля сообщения для текущего сообщения. Содержимое и формат данного поля идентичны содержимому и формату поля "Поля сообщения, принятые по умолчанию";

параметр FAO -

Длинное слово, значение которого используется директивой FAO, появляющейся в тексте сообщения. Параметры FAO, перечисленные в дескрипторе сообщения, должны появляться в том порядке, в котором они будут использоваться директивами FAO из текста сообщения.

Дескриптор сообщения для системных сообщений показан на рис.75.

Дескриптор сообщения для системных сообщений

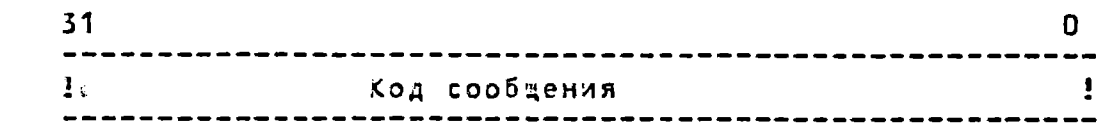


Рис. 75

Дескриптор сообщения для сообщений СУД-32 показан на рис.76.

Дескриптор сообщения для сообщений СУД-32

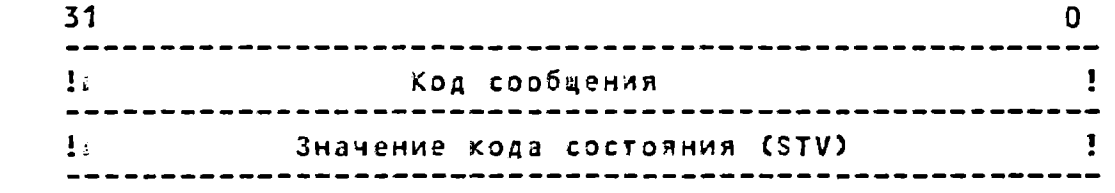


Рис. 76

Поля дескриптора сообщения для сообщений СУД-32:

код сообщения -

Длинное слово, значение которого уникально идентифицирует сообщение. Поле номера средства в коде сообщения идентифицирует средство, связанное с сообщением. Сообщение системы СУД-32 имеет номер средства 1. Поле "Счетчик параметров FAO", "Новые поля сообщения" и "Параметр FAO" не указываются. Длинное слово, следующее за полем "Код сообщения" в векторе сообщения, будет интерпретироваться как STV;

значение кода состояния системы СУД-32 -

Длинное слово, содержащее STV для использования сообщением СУД-32, которое имеет связанное значение STV. Программа $PUTMSG использует STV в качестве параметра

FAO или в качестве другого идентификатора сообщения, в зависимости от сообщения СУД-32, идентифицированного полем "Код сообщения". Если сообщение СУД-32 не имеет связанного STV, то программа $PUTMSG игнорирует длинное слово STV, содержащееся в дескрипторе сообщения.

Дескриптор сообщения для сообщения об исключительных состояниях системы показан на рис.77.

Дескриптор сообщения для сообщения об исключительных ситуациях системы

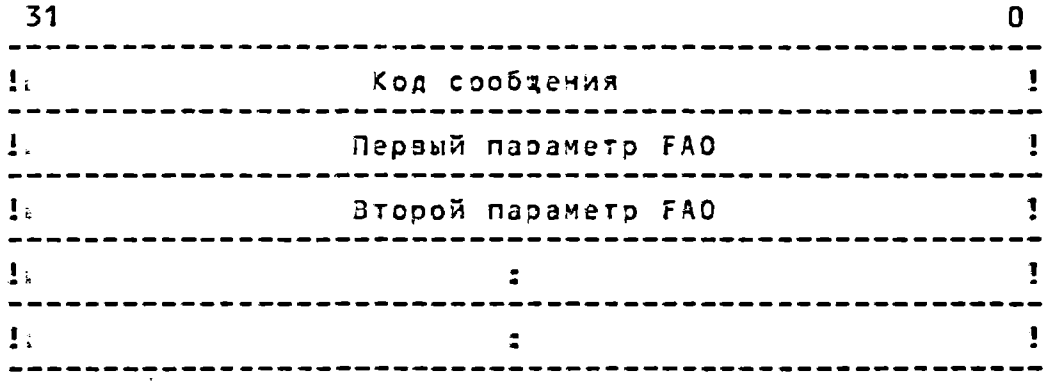


Рис. 77

Поля в дескрипторе сообщения для сообщений об исключительных состояниях системы:

код сообщения -

Длинное слово, значение которого уникально идентифицирует сообщение. Поле "Номер средства" в коде сообщения идентифицирует средство, связанное с сообщением. Сообщение об исключительных состояниях системы имеет номер средства 0. Поля "Счетчик параметров FAO" и "Новые поля сообщения" не указываются. Длинные слова, следующие за полем "Код сообщения" в векторе сообщения,

будут интерпретироваться как параметры FAO.

ACTRTN

Использование в MOC ВП: PROCEDURE

Тип: маска входа в программу

Доступ: вызов без развертывания стека

Механизм: по ссылке

Программа обработки, предоставленная пользователем, которая должна выполняться во время обработки сообщения. Аргумент ACTRTN является адресом маски входа в данную программу.

Программа обработки получает управление после того, как сообщение сформировано, но перед тем, как оно реально поступает пользователю.

Если аргумент ACTRTN не задан или задан равным 0 (данное значение по умолчанию), то не выполняется никаких программ обработки.

Поскольку программа $PUTMSG записывает сообщение только в SYS$ERROR и SYS$OUTPUT, то программа обработки полезна, например, когда вывод должен быть направлен в файл.

FACNAM

Использование в MOC ВП: CHAR_STRING

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Префикс средства, который должен использоваться в первом или единственном сообщении, записанном программой

$PUTMSG. Аргумент FACNAM является адресом дескриптора строки символов, указывающего на данный префикс средства.

Если арумент FACNAM не задан, то программа $PUTMSG использует принятый по умолчанию префикс средства, который связан с данным средством.

ACTPRM

Использование в МОС ВП: USER_ARG

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Параметр, который должен передаваться программе обработки. Аргумент ACTPRM является длинным словом, значение которого содержит данный параметр. Если аргумент ACTPRM не задан, то никакие параметры не передаются.

Первым параметром, передаваемым программе обработки, является адрес дескриптора строки символов, указывающий на текст сообщения, а параметр, задаваемый аргументом ACTPRM, передается вторым.

Описание

В операционной системе МОС ВП сообщение идентифицируется значением размером в длинное слово, которое называется "Кодом сообщения". Чтобы сконструировать код сообщения, необходимо задать значения четырех его полей с помощью программы MESSAGE. Рис.78 представляет длинное слово, содержащее код сообщения.

Код сообщения

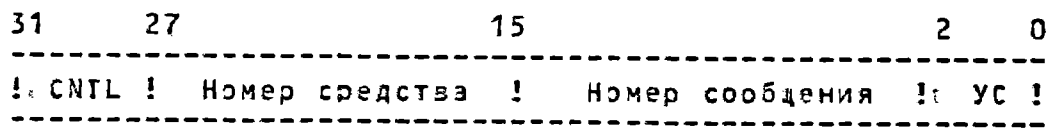


Рис. 78

Таким образом, каждое сообщение имеет связанное с ним уникальное значение размером в длинное слово - код данного сообщения. Можно дать данному значению символическое имя. Такое символическое имя называется "символическим именем сообщения".

Компоненты сообщения, которые записываются программой ¤PUTMSG, формируются на основе как кода сообщения, так и символического имени сообщения.

Программа системного обслуживания ¤PUTMSG пишет компонетны сообщения в следующем формате:

%средство - УС - идент, текст сообщения

где % - префикс, используемый для первого записанного сообщения. Для остальных сообщений в качестве префикса используется дефис (-);

средство - префикс средства, взятый из символического имени сообщения. Данный префикс средства может быть заменен префиксом средства, который задан аргументом FACNAM при вызове программы ¤PUTMSG;

УС - индикатор уровня серьезности сообщения. Индикатор уровня серьезности берется из кода сообщения;

идент - мнемоническое имя для текста сообщения. Данный

сегмент берется из символического имени сообщения;

текст сообщения - текст сообщения определяется в исходном файле сообщений.

Программа системного обслуживания ¤PUTMSG не проверяет длину списка аргументов и, следовательно, не может возвращать значение кода состояния SS¤_INSFARG (недостаточно аргументов). Пользователь должен сам проследить за тем, чтобы при вызове программы было указано необходимое число аргументов.

Если ошибка возникает при вызове из программы ¤PUTMSG программы системного обслуживания "форматировать вывод" (¤FAO), то параметры FAO, заданные в векторе сообщения, не появятся на выходе.

Программу системного обслуживания ¤PUTMSG нельзя вызывать из режима ядра.

Возвращаемое значение кода состояния:

SS¤_NORMAL - программа системного обслуживания завершилась нормально.

13.82. ¤QIO - поставить в очередь запрос на ввод-вывод

Программа системного обслуживания ¤QIO ставит в очередь запрос на ввод-вывод в очередь к каналу, связанному с устройством.

Программа системного обслуживания ¤QIO завершается асинхронно, то есть она возвращает управление вызывающему

процессу сразу же после постановки в очередь запроса на ввод-вывод, не ожидая, пока закончится операция ввода-вывода.

Для синхронного завершения, следует использовать программу системного обслуживания "поставить в очередь запрос на ввод-вывод и ждать" ($QIOW). Программа системного обслуживания $QIOW идентична программе системного обслуживания $QIO во всем, кроме того факта, что программа $QIOW возвращает управление вызывающему процессу после завершения операции ввода-вывода (см. подраздел 2.5).

Формат:

SYS$QIO [EFN],CHAN,FUNC,[IOSB],[ASTADR],[ASTPRM]
,[P1],[P2],[P3],[P4],[P5],[P6]

Аргументы:

EFN

Использование в МОС ВП: EF_NUMBER

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Флаг события, который устанавливается программой $QIO после действительного завершения операции ввода-вывода. Аргумент EFN является длинным словом, содержащим номер данного флага события.

Если аргумент EFN не задан, то устанавливается флаг события 0.

Когда программа $QIO начинает выполнение, она сбрасывает указанный флаг события или флаг события 0, если аргу-

мент EFN не задан.

Указанный флаг события устанавливается, если программа системного обслуживания завершается, не поставив в очередь запрос на ввод-вывод.

CHAN

Использование в МОС ВП: CHANNEL

Тип: слово (без знака)

Доступ: только чтение

Механизм: по значению

Канал ввода-вывода, назначаемый устройству, на которое поставлен запрос. Аргумент CHAN является длинным словом, значение которого содержит номер канала вода-вывода; однако программа $QIO использует только младшую часть данного слова.

FUNC

Использование в МОС ВП: FUNCTION_CODE

Тип: слово (без знака)

Доступ: только чтение

Механизм: по значению

Коды функции специфичные для конкретных устройств и модификаторы функции, задающие, какая операция должна выполняться. Аргумент FUNC является длинным словом, содержащим код функции.

Каждое устройство имеет свои коды функции и свои модификаторы функции. В документе [2] приводится полная информация о функциональных кодах и модификаторах функций, относящихся к конкретному устройству, на которое должна быть

направлена данная операция ввода-вывода.

IOSB

Использование в МОС ВП: IO_STATUS_BLOCK

Тип: квадрослово

Доступ: только запись

Механизм: по ссылке

Блок состояния ввода-вывода, в который заносится конечное состояние завершения операции ввода-вывода. Аргумент IOSB является адресом квадрослова, содержащего блок состояния ввода-вывода. Формат IOSB изображен на рис.79.

Формат блока IOSB

31	15	0
Счетчик передачи	Значение кода состояния	
Информация, специфичная для устройства		

Рис. 79

Поля блока состояния ввода-вывода:

значение кода состояния -

Значение кода состояния размером в слово, возвращаемое программой $QIO после действительного завершения операции ввода-вывода;

счетчик передачи -

Число байтов данных, действительно перемещенных при операции ввода-вывода;

информация, зависящая от устройства -

Содержимое данного поля меняется в зависимости от конкретного устройства и заданного кода функции. В

документе [2] приведена информация о том, как конкретные устройства управляют данным полем состояния ввода-вывода.

Когда программа ¤QIO начинает выполняться, она очищает квадрослово блока состояния ввода-вывода, если аргумент задан.

Хотя IOSB является необязательным аргументом, рекомендуется использовать его по следующим причинам:

1) если для сигнализации о завершении данной программы системного обслуживания используется флаг события, то можно проверить в блоке состояния ввода-вывода значение кода состояния, чтобы удостовериться, что данный флаг события не был установлен каким-либо событием, отличным от завершения программы системного обслуживания;

2) если для синхронизации данной программы системного обслуживания используется программа системного обслуживания ¤SYNCH, то блок состояния ввода-вывода является обязательным аргументом для программы ¤SYNCH;

3) значение кода состояния, возвращаемое в регистре R0, и значение кода состояния, возвращаемое в блоке ввода-вывода, предоставляют информацию о разных аспектах обращения к программе ¤QIO. Значение кода состояния, возвращаемое в регистре R0, дает информацию об успешном или неудачном вызове программы системного обслуживания. Следовательно, чтобы иметь точное представление об успешном или неудачном обращении к программе ¤QIO, необходимо проверить значения кода состояния, возвращаемые как в регистре R0,

так и в блоке состояния ввода-вывода.

ASTADR

Использование в МОС ВП: AST_PROCEDURE

Тип: маска входа в программу

Доступ: вызов без развертывания стека

Механизм: по ссылке

Подпрограмма обработки прерывания AST, которая должна выполняться после завершения ввода-вывода. Аргумент ASTADR является адресом длинного слова, значение которого есть маска входа в подпрограмму AST.

Подпрограмма AST выполняется в режиме доступа процесса, вызвавшего программу $QIO.

ASTPRM

Использование в МОС ВП: USER_ARG

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Параметр AST, который должен передаваться в подпрограмму обработки прерывания AST. Аргумент ASTPRM является длинным словом, значение которого содержит параметр AST.

С P1 по P6

Использование в МОС ВП: VARYING_ARG

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Необязательные параметры запроса на ввод-вывод, специфичные для конкретного устройства и функции.

Первый параметр может быть задан как P1 или P1V. Если функциональный код требует адреса, следует задавать P1, если он требует значения, то следует задавать P1V. Если при вызове программы $QIO не указано ключевое слово P1 или P1V, то предполагается параметр P1, означающий, что аргумент рассматривается как адрес.

Ключевые слова с P2 по P6 всегда интерпретируются как значения.

Описание

Программа $QIO работает только с назначенными каналами ввода-вывода и только из тех режимов доступа, которые равны или являются более привилегированными, чем режим доступа, из которого было сделано первоначальное назначение канала.

Программа $QIO использует следующие системные ресурсы:

1) квоту процесса для лимита ввода-вывода с буферизацией (BIOLM) или лимита прямого ввода-вывода (DIOLM);

2) квоту счетчика байтов ввода-вывода с буферизацией (BYTLM);

3) квоту лимита AST процесса (ATSLM), если задана какая-либо подпрограмма обработки AST;

4) может потребоваться дополнительная память в зависимости от конкретного устройства.

Для программы $QIO завершение можно синхронизировать двумя способами:

1) указав аргумент ASTADR, чтобы выполнить подпрограмму AST после завершения ввода-вывода;

2) вызвав программу системного обслуживания $SYNCH,

что позволит ожидать завершения операции ввода-вывода. Программа $QIOW завершается синхронно и это лучший способ, если требуется синхронное завершение.

Возвращаемые значения кода состояния:

SS$_NORMAL - программа системного обслуживания завершилась успешно. Запрос на ввод-вывод успешно поставлен в очередь;

SS$_ABORT - разорвана логическая связь в сети;

SS$_ACCVIO - либо нельзя сделать запись в блок состояния ввода-вывода, либо некорректно заданы параметры для кодов функций, зависящих от устройств;

SS$_DEVOFFLINE - указанное устройство находится в автономном состоянии и в текущий момент недоступно для использования;

SS$_EXQUOTA - процесс либо превысил свою квоту лимита AST (ASTLM); либо превысил свою квоту счетчика байтов ввода-вывода с буферизацией (BYTLM); либо превысил свою квоту лимита ввода-вывода с буферизацией (BIOLM); либо превысил свою квоту прямого ввода-вывода (DIOLM); либо запрос для ввода-вывода с буферизацией меньше лимита счетчика байтов с буферизацией (BYTLM), но будучи добавленным к дру-

гим текущим запросам буфера, вызвал превышение карты счетчика байтов буферизованного ввода-вывода;

SS$_ILLEFC - задан неверный номер флага события;

SS$_INSFMEM - для завершения данной программы системного обслуживания нет доступной динамической памяти достаточного размера;

SS$_IVCHAN - указан неверный номер канал, то есть номер канала 0 или номер больший, чем число доступных каналов;

SS$_NOPRIV - указанный канал либо не существует, либо был назначен из более привилегированного режима доступа; или процесс не имеет необходимых привилегий для выполнения заданных функций на устройстве, связанном с указанным каналом;

SS$_UNASEFC - процесс не связан с кластером, содержащим указанный флаг события;

- SS$_LINKABORT задача партнера по сети прервала логическую связь;

SS$_LINKDISCON - задача партнера по сети отсоединила логическую связь;

SS$_PATHLOST - утерян путь к узлу задачи партнера по сети;

SS$_PROTOCOL - произошла ошибка протокола сети.

вероятнее всего это произошло из-за ошибки программного обеспечения сети;

SS$_CONNECFAIIL - соединение с об'ектом сети было отключено из-за превышения лимита времени или произошел сбой;

SS$_FIALRACC - к логической связи уже получен доступ в канале (то есть, по предыдущему соединению в канале);

SS$_INVLOGIN - в удаленном узле обнаружена неверная информация управления досупом;

SS$_IVDEVNAM - блок NCB имеет неверный формат или содержимое;

SS$_LINKEXIT - была запущена задача партнера по сети, но выход произошел без подтверждения логической связи (то есть ASSIGN в SYS$NET);

SS$_NOLINKS - нет доступных логических связей. Превышено максимальное число логических связей, которое было установлено для управляющей программы параметром MAXIMUMLINKS;

SS$_NOSUCHNODE - указанный узел не известен системе;

SS$_NOSUCHOBJ - номер об'екта сети неизвестен в удаленном узле; или в удаленном узле не найден названный программный файл DCL для команды TASK=CONNECT;

SS$_NOSUCHUSER - удаленный узел не смог распознать

информацию, заданную при подключении к системе (LOGIN);

SS$_PRTOCOL - произошла ошибка протокола. Вероятнее всего это произошло из-за ошибки программного обеспечения сети;

SS$_REJECT - об'ект сети отверг соединение;

SS$_REMRSRC - связь не может быть установлена из-за недостаточных системных ресурсов в удаленном узле;

SS$_SHUT - локальный или удаленный узел больше не принимает соединений;

SS$_THIRDPARTY - логическая связь была завершена третьей стороной (например, администратором системы);

SS$_TOOMUCHDATA - задача показала слишком много необязателных данных или данных прерывания;

SS$_UNREACHABLE - удаленный узел в текущий момент недоступен.

Значения кода состояния, возвращаемые в блоке состояния ввода-вывода:

В документе [2] перечислены значения кодов состояния для каждого устройства в разделе, описывающем данное устройство.

13.83. $QIOW - поставить в очередь запрос на ввод-вывод и ждать

Программа системного обслуживания $QIOW ставит в очередь запрос на ввод-вывод в очередь к каналу, связанному с устройством.

Программа системного обслуживания $QIOW завершается синхронно, то есть она возвращает управление вызывающему процессу после действительного завершения операции ввода-вывода.

Для асинхронного завершения следует использовать программу системного обслуживания "поставить в очередь запрос на ввод-вывод" ($QIO), которая возвращает управление вызывающему процессу сразу же после постановки в очередь запроса на ввод-вывод, не ожидая завершения операции ввода-вывода.

Во всех остальных отношениях программы $QIOW и $QIO идентичны. Всю информацию, касающуюся программы системного обслуживани $QIO можно использовть для програмы $QIOW (см. подраздел 2.5).

Формат:

SYS$QIOW [EFN],CHAN,FUNC[,IOSB][,ASTADR]
[,ASTPRM][,P1][,P2][,P3][,P4][,P5][,P6]

13.84. ¤READEF - читать флаги событий

Программа системного обслуживания ¤READEF возвращает текущее состояние всех 32 флагов событий в кластере локальных или общих флагов событий. Кроме того, возвращаемое значение кода состояния индицирует, был ли установлен или сброшен указанный флаг события.

Формат:

SYS¤READEF EFN,STATE

Аргументы:

EFN

Использование в МОС ВП: EF_NUMBER

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Номер любого флага события в кластере, состояние которого должна вернуть программа. Аруметт EFN является длинным словом, содержащим данный номер. Указание флага события внутри кластера требует, чтобы программа ¤READEF вернула состояния всех флагов событий в данном кластере.

Имеется два кластера локальных флагов событий, которые являются локальными для процесса - кластер 0 и кластер 1. Кластер 0 содержит флаги событий с номерами с 0 по 31, а кластер событий 1 содержит флаги событий с номерами с 32 по 63.

Имеется два кластера общих флагов событий - кластер 2 и кластер 3. Кластер 2 содержит флаги событий с номерами с 64 по 95, а кластер 3 содержит флаги событий с номерами с

96 по 127.

STATE

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Состояние всех флагов событий в указанном кластере. Аргумент STATE является адресом длинного слова, в которое программа ¤READEF записывает состояние (установлен или нет) всех 32 флагов событий в кластере.

Возвращаемые значения кода состояния:

SS¤_WASCLR - программа системного обслуживания успешно завершилась. Указанный флаг события сброшен;

SS¤_WASSET - программа системного обслуживания завершилась. Указанный флаг события установлен;

SS¤_ACCVIO - вызывающий процесс не может записать в длинное слово, которое должно принимать текущее состояние всех флагов событий;

SS¤_ILLEFC - указан неверный номер флага события;

SS¤_UNASEFC - процесс не связан с кластером, содержащим указанный флаг события.

13.85. ¤REM_HOLDER - удалить запись держателя из базы данных прав

Программа системного обслуживания ¤REM_HOLDER удаляет указанную запись держателя из списка держателей соответст-

вующего идентификатора.

Формат:

SYS$REM_HOLDER ID,HOLDER

Аргументы:

ID

Использование в МОС ВП: RIGTHS_ID

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Двоичное значение соответсвующего идентификатра, чей держатель удаляется в результате работы программы системного обслуживания $REM_HOLDER. Аргумент ID является длинным словом, содержащим значение идентификатора.

HOLDER

Использование в МОС ВП: RIGHTS_HOLDER

Тип: квадрослово (без знака)

Доступ: только чтение

Механизм: по ссылке

Идентификатор держателя, который удаляется в результате работы программы системного обслуживания $REM_HOLDER. Аргумент HOLDER является адресом квадрослова, содержащего идентификатор UIC держателя в длинном слове и нулевое значение во втором длинном слове.

Описание

Программа системного обслуживания "удалить запись держателя из базы данных прав" удаляет запись указанного держателя из списка держателей соответствующего идентификатора.

Для использования данной программы системного обслуживания требуется доступ с записью в базу данных прав. Если база данных находится в группе файлов SYS¤SYSTEM (что принимается по умолчанию), то для получения доступа с записью в базу данных прав требуется привилегия SYSPRV.

Возвращаемые значения кодов состояния:

SS¤_NORMAL - успешное завершение;

SS¤_ACCVIO - вызывающий процесс не может прочитать аргументы ID или HOLDER;

SS¤_INSFMEM - для открытия базы данных прав нет доступной динамической памяти процесса достаточного размера;

SS¤_IVIDENT - указанный идентификатор или идентификатор держателя имеет неверный формат;

SS¤_NOSUCHID - указанного идентификатора нет в базе данных прав; или указанного идентификатора держателя нет в базе данных прав;

RMS¤_PRV - пользователь не имеет доступа с записью в базу данных прав.

Поскольку база данных прав является индексным файлом, к которому осуществляется с помощью системы управления данными СУД-32, данная программа системного обслуживания может

также возвращать коды состояния системы СУД-32, связанные с операциями на индексном файле (см. документ [1]).

13.86. ¤REM_IDENT - удалить идентификатор из базы данных прав

Программа системного обслуживания ¤REM_IDENT удаляет запись указанного идентификатора и записи всех его держателей (если таковые имеются) из базы данных прав.

Формат:

SYS¤REM_IDENT ID

Аргументы:

ID

Использование в МОС ВП: RIGHTS_ID

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Двоичное значение идентификатора, удаляемого из базы данных прав в результате работы программы ¤REM_IDENT. Аргумент ID является длинным словом, содержащим данное значение идентификатора.

Описание.

Программа системного обслуживания ¤REM_IDENT удаляет указанный идентификатор из базы данных прав. Все записи держателей, связанные с данным идентификатором, также удаляются. Кроме того, удаляются все записи в идентификаторах, которые удерживал удаляемый идентификатор.

Для использования данной программы системного обслуживания требуется доступ с записью в базу данных прав. Если база данных прав находится в группе файлов SYS¤SYSTEM (что принимается по умолчанию), то для получения доступа к данной базе данных прав требуется привилегия SYSPRV.

Возвращаемые значения кода состояния:

SS¤_NORMAL - программа системного обслуживания успешно завершилась;

- SS¤_INSFMEM для открытия базы данных прав нет доступной динамической памяти процессора достаточного размера;
- SS¤_IVIDENT указанный идентификатор имеет неверный формат;
- SS¤_NOSUCHID указанный идентификатор отсутствует в базе данных прав;
- RMS¤_PRV пользователь не имеет доступа с записью в базу данных прав.

Поскольку база данных прав является индексным файлом, доступ к которому осуществляется с помощью системы управления данными СУД-32, данная программа системного обслуживания может также возвращать коды состояния системы СУД-32, связанные с операциями на индексном файле (см. документ [1]).

13.87. $RESUME - возобновить процесс

Программа системного обслуживания $RESUME заставляет процесс, ранее приостановленный программой системного обслуживания "приостановить процесс" ($SUSPND), возобновить выполнение, или отменяет действие последующего запроса на приостановку процесса.

Формат:

SYS$RESUME [PIDADR],[PRCNAM]

Аргументы:

PIDADR

Использование в МОС ВП: PROCESS_ID

Тип: длинное слово (без знака)

Доступ: модификация

Механизм: по ссылке

Идентификатор процесса (PID), который нужно возобновить. Аргумент PIDADR - это адрес длинного слова, содержащего PID.

Аргумент PIDADR нужно задавать для возобновления процессов в других группах UIC.

PRCNAM

Использование в МОС ВП: PROCESS_NAME

Тип: строка текста

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки с фиксированной длиной

Имя процесса, который нужно возобновить. Аргумент PRCNAM - это адрес дескриптора символьной строки, указываю-

щего на имя процесса, которое представляет символьную строку длиной от 1 до 15 символов.

Аргумент PRCNAM можно использовать для возобновления только тех процессов, которые принадлежат той же группе UIC, что и вызывающий процесс. Причина заключается в том, что имена процессов уникальны в пределах групп UIC, и мос вл. использует номер группы UIC при интерпретации имени процесса, заданного аргументом PRCNAM. Для возобновления процессов в других группах UIC нужно использовать аргумент PIDADR.

Если не заданы ни аргумент PIDADR, ни аргумент PRCNAM, то запрос на возобновление выдается для самого вызывающего процесса.

Описание.

В зависимости от конкретной операции использование программы $RESUME может потребовать наличия у вызывающего процесса определенных привилегий:

1) привилегия GROUP нужна для того, чтобы возобновить выполнение процесса в той же группе, если данный процесс имеет UIC, отличный от UIC вызывающего процесса;

2) привилегия WORLD нужна для того, чтобы возобновить выполнение любого процесса в системе.

Если для процесса, который не приостановлен, выданы один или несколько запросов на возобновление, то последующий запрос на приостановку завершится сразу же, то есть процесс не приостанавливается. Для выставленных запросов на возобновление счетчик не ведется.

Возвращаемые значения кода состояния:

SS¤_NORMAL - успешное завершение;

SS¤_ACCVIO вызывающая строка не может прочитать строку имени процесса или дескриптор строки, либо не может записать идентификацию процесса;

- SS¤_IVLOGNAM заданное имя процесса имеет нулевую длину или длину более 15 символов;
- SS¤_NONEXPR предупреждение. Указанный процесс не существует или была задана неверная идентификация;
- SS¤_NOPPIV у процесса нет привилегии для возобновления выполнения указанного процесса.

13.88. ¤REVOKID - исключить идентификатор из процесса

Программа системного обслуживания ¤REVOKID удаляет из списка прав процесса или системы указанный идентификатор. Если данный идентификатор значится в списке как держатель какого-то другого идентификатора, то соответствующие записи держателя также удаляются.

Формат:

SYS¤REVOKID [PIDARD],[PRCNAM],[ID],[NAME],[PRVATR]

Аргументы:

PIDARD

Использование в МОС ВП: PROCESS_ID

Тип: длинное слово (без знака)

Доступ: модификация

Механизм: по ссылке

Идентификатор процесса (PID) того процесса, с которым работает программа $REVOKID. Аргумент PIDARD является адресом длинного слова, содержащего идентификатор PID данного процесса. Чтобы указать системный список прав, следует задавать значение аргумента PIDARD равным минус 1. Когда аргумент PIDARD передается программе, он также и возвращается программой, следовательно необходимо передавать его как переменную, а не как константу.

PRCNAM

Использование в МОС ВП: PROCESS_NAME

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Имя процесса, с которым работает программа $REVOKID. Аргумент PRCNAM является адресом дескриптора строки символов, содержащего имя процесса. Максимальная длина данного имени составляет 15 символов. Поскольку групповой номер UIC интерпретируется как часть имени процесса, то для того, чтобы указать список прав процесса из другой группы, следует использовать аргумент PIDADR.

ID

Использование в МОС ВП: RIGHTS_ID

Тип: квадрослово (без знака)

Доступ: модификация

Механизм: по ссылке

Идентификатор и атрибуты, которые удаляются в результате работы программы ¤REVOKID. Аргумент ID является адресом квадрослова, содержащего в первом длинном слове, двоичный код идентификатора, а во втором длинном слове - атрибуты.

В системной макробиблиотеке (макрокоманда ¤KGBDEF) определено символическое имя:

1) KGB¤V_RESOURCE - позволяет держателю запрашивать для идентификатора ресурсы типа блоков на дисках.

Символическое значение является смещением в битах внутри длинного слова. Данное значение можно также получать в виде маски с установленным соответствующим битом, если вместо префикса KGB¤V использовать префикс KGB¤M.

Необходимо задать один из аргументов ID или NAME. Поскольку аргумент ID возвращается программой, если задан аргумент NAME (а также, если задан сам аргумент ID), то в данном случае необходимо передавать аргумент ID как переменную, а не как константу.

NAME

Использование в МОС ВП: CHAR_STRING

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Имя идентификатора, который удаляется в результате работы программы $REVOKID. Аргумент NAME является адресом дескриптора, указывающего на имя идентификатора.

PRVATR

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Атрибуты удаляемого идентификатора. Аргумент PRVATR является адресом длинного слова, которое используется для хранения атрибутов идентификатора.

Описание:

Поскольку программа системного обслуживания "исключить идентификатор из процесса" удаляет указанный идентификатор из списка прав процесса или системного списка прав, данная программа системного обслуживания предназначена для использования привилегированной подсистемой для изменения профиля прав пользователя на доступ на основе установочной стратегии. Она не предназначена для использования обычными пользователями операционной системы МОС ВП.

Для вызова данной программы системного обслуживания требуется привилегия CMKRNL. Кроме того для модификации списка прав процесса, принадлежащего той же группе, что и вызывающий процесс, требуется привилегия GROUP (если только процесс не имеет тот же идентификатор UIC, что и вызывающий

процесс). Для модификации списка прав процесса вне группы вызывающего процесса требуется привилегия WORLD. Для модификации системного списка прав требуется привилегия SYSNAM.

Результаты передачи аргументов PIDADR и PRCNAM программе $REWOKID приведены в табл.66.

Таблица 66

PRCNAM	PIDADR	Результат
опущен	опущен	используется идентификатор текущего процесса. Идентификатор процесса не возвращается
опущен	0	используется идентификатор текущего процесса. Идентификатор процесса возвращается
опущен	задан	используется заданный идентификатор процесса. Идентификатор процесса возвращается
задан	опущен	используется заданное имя процесса. идентификатор процесса не возвращается
задан	0	используется заданное имя процесса. Идентификатор процесса возвращается
задан	задан	используется заданный идентификатор процесса. Идентификатор процесса возвращается. Имя процесса игнорируется

Результаты передачи аргументов ID и NAME программе ¤REVOKID приведены в табл.67.

Таблица 67

NAME	ID	Результат
опущен	опущен	недопустимо
опущен	задан	используется заданное значение идентификатора. Значение идентификатора возвращается
задан	опущен	используется заданное имя идентификатора. Значение идентификатора не возвращается
задан	0	используется заданное имя идентификатора. Значение идентификатора возвращается
задан	задан	используется заданное значение идентификатора. Значение идентификатора возвращается. Имя идентификатора игнорируется

Возвращаемые значения кода состояния:

SS¤_WASCLR - программа системного обслуживания успешно завершилась; список прав не содержал указанного идентификатора;

SS¤_WASSET - программа системного обслуживания успешно завершилась; список прав содержал указанный идентификатор;

SS$_ACCVIO - невозможно прочитать или записать аргумент PIDADR; невозможно прочитать аргумент PRCNAM; невозможно прочитать или записать аргумент ID; невозможно прочитать аргумент NAME; невозможно записать аргумент PRVATR;

SS$_INSFMEM - нет доступной динамической памяти процесса достаточного размера для открытия базы данных прав;

SS$_NOPRIV - у вызывающего процесса нет привилегии CMKRNL; или он не выполняется в режиме ядра или управления; или у вызывающего процесса нет требуемой привилегии GROUP, WORLD или SYSNAME;

SS$_NOSUCHID - указанное имя идентификатора отсутствует в базе данных прав. Если задан двоичный идентификатор, то он не проверяется в базе данных прав;

SS$_RIGHTSFULL - список прав процесса или системы полон;

SS$_IVIDENT - указанный идентификатор или держатель имеет неверный формат; или указанные идентификатор и держатель идентичны;

SS$_NOSYSNAME - операция требует привилегии SYSNAME;

SS$_IVLOGNAM - задано неверное логическое имя;

SS$_NONEXPR - задан несуществующий процесс;

RMS$_PRV - пользователь не имеет доступа с чтением в базу данных прав.

Поскольку база данных прав является индексным файлом, доступ к которому осуществляется с помощью системы управления данными СУД-32, данная программа системного обслуживания может также возвращать коды состояния системы СУД-32, связанные с операциями на индексном файле (см. документ [1]).

13.39. $SCHDWK - запланировать пробуждение

Программа системного обслуживания $SCHDWK планирует пробуждение процесса, который перевел себя в состояние спячки при помощи программы системного обслуживания "перевести в состояние спячки" ($HIBER). Пробуждение можно запланировать на заданное абсолютное время или через дельта-время, и можно повторять через фиксированные интервалы времени.

Формат:

SYS$SCHDWK [PIDADR],[PRCNAM],DAYTIM,[REPTIM]

Аргументы:

PIDADR

Использование в МОС ВП: PROCESS_ID

Тип: длинное слово (без знака)

Доступ: модификация

Механизм: по ссылке

Идентификатор процесса (PID), который требуется пробудить. Аргумент PIDADR - это адрес длинного слова, содержа-

щего PID.

Аргумент PIDADR нужно указывать для пробуждения процессов в других группах UIC.

PRCNAM

Использование в МОС ВП: PROCESS_NAME

Тип: строка текста

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки с фиксированной длиной

Имя процесса, который нужно пробудить. Аргумент PRCNAM - это адрес дескриптора символьной строки, указывающего на имя процесса, которое представляет символьную строку длиной от 1 до 15 символов.

Аргумент PRCNAM можно использовать для пробуждения только тех процессов, которые принадлежат к той же группе UIC, что и вызывающий процесс. Причина заключается в том, что имена процессов уникальны в пределах группы UIC, и мос вп использует номер группы UIC вызывающего процесса при интерпретации имени процесса, заданного аргументом PRCNAM. Для пробуждения процессов в других группах UIC необходимо использовать аргумент PIDADR.

Если не заданы ни аргумент PIDADR, ни аргумент PRCNAM, то запрос на пробуждение выдается для самого вызывающего процесса.

DAYTIM

Использование в МОС ВП: DATE_TIME

Тип: квадрослово (без знака)

Доступ: только чтение

Механизм: по ссылке

Время, в которое нужно пробудить процесс. Аргумент DAYTIM - это адрес квадрослова, содержащего данное время в 64-битовом системном формате времени. Положительное значение времени задает абсолютный момент времени, в который нужно пробудить указанный процесс. Отрицательный момент времени задает смещение (дельта-время) от текущего времени.

REPTIM

Использование в МОС ВП: DATE_TIME

Тип: квадрослово (без знака)

Доступ: только чтение

Механизм: по ссылке

Интервал времени, через который нужно повторять запрос на пробуждение. Аргумент REPTIM - это адрес квадрослова, содержащего данный интервал. Интервал времени нужно представлять в формате дельта-времени.

Нельзя задавать интервал времени менее 10 миллисекунд; если он меньше 10 миллисекунд, $SCHDWK автоматически увеличивает его до 10 миллисекунд.

Если аргумент REPTIM не задан, по умолчанию используется значение 0; нуль означает, что запрос на пробуждение не нужно повторять.

Описание.

В зависимости от конкретной операции использование программы ¤SCHDWK может потребовать наличия у процесса определенных привилегий:

1) привилегия GROUP нужна для того, чтобы планировать запросы на пробуждение для процесса в той же группе, но с другим UIC;

2) привилегия WORLD нужна для того, чтобы планировать запросы на пробуждение для любого другого процесса в системе.

¤SCHDWK использует следующие системные ресурсы:

1) для планирования запроса на пробуждение используется квота предела AST (ASTLM) вызывающего процесса;

2) для назначения элемента в очереди к таймеру используется системная динамическая память.

Если для процесса, который не находится в состоянии спячки, выданы один или несколько запросов на пробуждение, то последующий запрос на спячку для данного процесса завершится сразу же, то есть процесс не переходит в состояние спячки. Для востановленных запросов на пробуждение не ведется счетчик.

Запланированные запросы на пробуждение, если они еще не обработаны, можно отменить при помощи программы системного обслуживания "отменить пробуждение" (¤CANWAK).

Если заданное абсолютное время уже прошло и время повторения не указано, то таймер срабатывает в следующий цикл часов (в пределах 10 миллисекунд).

Возвращаемые значения кода состояния:

SS$_NORMAL - успешное завершение;

- SS$_ACCVIO вызывающая программа не может прочесть время окончания, время повторения, строку имени процесса или дескриптор строки, либо не может записать идентификатор процесса;

- SS$_EXQUOTA процесс превысил свою квоту предел;

- SS$_INSFMEM не хватает системной динамической памяти для назначения элемента в очереди к таймеру;

- SS$_IVLOGNAM строка имени процесса имеет нулевую длину или длину более 15 симзолов;

- SS$_IVTIME заданное дельта-время повторения является положительной величиной, либо абсолютное время плюс дельта-время повторения меньше текущего времени;

- SS$_NONEXPR предупреждение. Указанный процесс не существует или был задан неверный идентификатор процесса.

SS$_NOPRIV у процесса нет полвилегии на планирование запросов пробуждения для указанного процесса.

13.90. ¤SETAST - установить AST

Программа системного обслуживания ¤SETAST включает или выключает доставку прерываний AST для режима доступа, из которого был сделан вызов данной программы системного обслуживания.

Формат:

SYS¤SETAST ENBFLG

Аргументы:

ENBFLG

Использование в МОС ВП:	BOOLEAN
Тип:	байт (без знака)
Доступ:	только чтение
Механизм:	по значению

Значение, указывающее, должна ли быть включена доставка AST. Аргумент ENBFLG является байтом, содержащим данное значение. Значение 1 включает доставку AST для вызывающего режима доступа, а значение 0 выключает доставку AST.

Описание.

Если образ выполняется в режиме пользователя, то доставка AST включается для всех более привилегированных режимов доступа.

Если доставка AST отключается для более привилегированного режима доступа, то операционная система МОС ВП не может доставлять прерывания AST для менее привилегированных режимов доступа до тех пор, пока доставка AST не будет снова включена для более привилегированного режима доступа. Следовательно, процесс, который отключил доставку прерыва-

ний AST для более привилегированного режима доступа, должен заново включить доставку AST для данного режима перед тем, как вернуться к менее привилегированному режиму доступа.

Возвращаемые значения кода состояния

- SS$_WASCLR программа системного обслуживания успешно завершилась доставка AST ранее была отключена для режима доступа вызывающего процесса;
- SS$_WASSET программа системного обслуживания успешно завершилась. Доставка AST ранее была включена для режима доступа вызывающего процесса.

13.91. $SETEF - установить флаг события

Программа системного обслуживания $SETEF устанавливает флаг события в кластере локальных или общих флагов событий. Значение кода состояния, возвращаемое программой $SETEF, индицирует, был ли ранее указанный флаг установлен или сброшен. После того как флаг события устанавливается, процессы, ожидающие данный флаг события, возобновляют выполнение.

Формат:

SYS$SETEF EFN

Аргументы:

EFN

Использование в МОС ВП: EF_NUMBER

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Номер флага события, который должен быть установлен в результате работы программы. Аргумент EFN является длинным словом, содержащим данный номер.

Имеется два кластера локальных флагов событий, которые локальны для процесса: кластер 0 и кластер 1. Кластер 0 содержит флаги событий с номерами от 0 до 31, а кластер 1 содержит флаги событий с номерами от 32 до 63.

Имеется два кластера общих флагов событий: кластер 2 и кластер 3. Кластер 2 содержит флаги событий с номерами от 64 до 95, а кластер 3 содержит флаги событий с номерами от 96 до 127.

Возвращаемые значения кода состояния:

SS$_WASCLR - программа системного обслуживания успешно завершилась. Указанный флаг события ранее был сброшен;

- SS$_WASSET программа системного обслуживания успешно завершилась; указанный флаг события ранеее был установлен;
- SS$_ILLEFC указан недопустимый номер флага события;
- SS$_UNASEFC процесс не связан с кластером, содержащим указанный флаг события.

13.92. ¤SETEXV - установить вектор исключительной ситуации

Программа системного обслуживания ¤SETEXV назначает адрес обработчика кода состояния для первичного, вторичного или окончательного вектора исключительной ситуации, или удаляет ранее назначенный адрес обработчика из любого из трех векторов.

Формат:

SYS¤SETEXV [VECTOR],[ADDRES],[ACMODE],[PRVHND]

Аргументы:

VECTOR

Использование в МОС ВП: LONGWORD_UNSIGNED

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Вектор, для которого надо организовать или удалить обработчик кода состояния. Аргумент VECTOR - это значение размером в длинное слово. Значение 0 (принимаемое по умолчанию) задает первичный вектор; значение 1 - вторичный вектор и значение 2 - окончательный вектор исключительной ситуации.

ADDRES

Использование в МОС ВП: PROCEDURE

Тип: маска входа программы

Доступ: вызов без развертывания стека

Механизм: по ссылке

Адрес обработчика кода состояния, организуемого для вектора исключительной ситуации, заданного аргументом VECTOR. Аргумент ADDRES - это значение длинного слова, содержащего адрес маски входа программы обработки кода состояния.

Если аргумент ADDRES не задан или равен 0, то адрес обработчика кода состояния, уже организованного для указанного вектора, удаляется из него, то есть содержимое вектора размером в длинное слово устанавливается в 0.

ACMODE

Использование в МОС ВП: ACCESS_MODE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Режим доступа, для которого надо модифицировать вектор исключительной ситуации. Аргумент ACMODE - это длинное слово, содержащее режим доступа. Макрокоманда $PSLDEF определяет обозначения для четырех режимов доступа.

Наивысший привилегированный режим доступа, который может использоваться - это режим доступа вызывающего процесса. Нельзя модифицировать векторы исключительной ситуации для режимов доступа, более привилегированных, чем режим доступа вызывающего процесса.

PRVHND

Использование в МОС ВП: PROCEDURE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Адрес предыдущего обработчика кода состояния, содержавшийся в указанном векторе исключительной ситуации. Аргумент PRVHND - это адрес длинного слова, в которое ¤SETEXV записывает адрес обработчика.

Описание.

Процесс не может модифицировать вектор, связанный с наиболее привилегированным режимом доступа.

МОС ВП обеспечивает два различных способа для организации обработчиков кода состояния:

1) использование стеков вызовов, связанного с каждым режимом доступа. Каждый кадр вызова включает длинное слово, которое должно содержать адрес обработчика кода состояния, связанного с данным кадром. Программа библиотеки исполнительных программ LIB¤ESTABLISH организует обработчик кода состояния; программа LIB¤REVERT удаляет обработчик;

2) использование векторов исключительной ситуации программного обеспечения (при помощи ¤SETEXV), связанных с каждым режимом доступа. Данные векторы устанавливаются в пределах области управления процесса (пространство P1).

Второй метод не обладает модульными возможностями, присущими первому методу. Векторы исключительной ситуации программного обеспечения предназначаются, в первую очередь, для мониторов производительности и отладчиков. Например, первичный вектор исключительной ситуации и окончательный вектор исключительной ситуации используются отладчиком мос вп для пользовательского режима доступа, а диалоговый

командный язык (DCL) использует окончательный вектор исключительной ситуации для супервизорного режима доступа.

Векторы исключительной ситуации пользовательского режима отменяются при осуществлении выхода образа.

Возвращаемые значения кода состояния:

SS¤_NORMAL - успешное завершение;

- SS¤_ACCVIO вызывающая программа не может записать длинное слово, предназначенное для получения прежнего содержимого вектора.

13.93. ¤SETIME - установить системное время

Программа системного обслуживания ¤SETIME изменяет значение системного времени или заново калибрует системное время.

Системное время определяется значением квадрослова, которое задает количество 100-наносекундных интервалов с 00 часов 17 ноября 1858.

Системное время используется почти всеми активными компонентами программного обеспечения в МОС ВП, связанными с таймером.

Формат:

SYS¤SETIME [TIMADR]

Аргументы:

TIMADR

Использование в МОС ВП: DATE_TIME

Тип: квадрослово (без знака)

Доступ: только чтение

Механизм: по ссылке

Новое значение системного времени. Аргумент TIMADR - это адрес квадрослова, содержащего новое значение системного времени. Новое значение системного времени - это значение абсолютного времени, задающее количество 100-наносекундных интервалов с 00 часов 17 ноября 1858. Отрицательное значение (дельта) времени не допускается.

Если аргумент TIMADR не указан или равен 0, то ¤SETIME заново калибрует системое время, используя часы времени года.

Описание:

Для того, чтобы установить системное время, вызывающий процесс должен иметь привилегии OPER и LOG_IO.

После изменения или повторной калибровки часов, ¤SETIME обновляет очередь к таймеру, подстраивая каждый элемент в очереди к таймеру с учетом разницы между прежним системным временем и новым системным временем.

Программа системного обслуживания ¤SETIME сохраняет новое время (для будущих операций перезагрузки) в системном образе SYS¤SYSTEM:SYS.EXE. Чтобы сохранить данное время, программа системного обслуживания назначает канал к устройству загрузки системы и вызывает программу системного

обслуживания $QIOW. Данная операция ввода-вывода требует привилегии LOG_IO.

Возвращаемые значения кода состояния:

SS$_NORMAL - успешное завершение;

- SS$_IVTIME вызывающий процесс не задал значение времени или задал отрицательное значение времени и обнаружилась некорректность процессорных часов;
- SS$_ACCVIO вызывающая программа не может прочитать квадрослово, содержащее значение нового системного времени;
- SS$_NOIOCHAN нет канала ввода-вывода, доступного для назначения;
- SS$_NOPRIV у процесса нет привилегии на установку системного времени.

13.94. $SETIMR - установить таймер

Программа системного обслуживания $SETIMR устанавливает таймер на срабатывание в заданный момент времени. Когда таймер срабатывает, устанавливается флаг события и выполняется (необязательно) подпрограмма обработки AST.

Формат:

SYS$SETIMR [EFN],DAYTIM,[ASTADR],[REQIDT]

Аргументы:

EFN

Использование в МОС ВП: EF_NUMBER

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Флаг события, который необходимо установить, когда срабатывает таймер. Аргумент EFN - это значение длинного слова, содержащего номер флага события. Если EFN не задан, то устанавливается флаг события 0.

Перед началом работы ¤SETIMR очищает заданный флаг события или флаг события 0.

DAYTIM

Использование в МОС ВП: DATE_TIME

Тип: квадрослово (без знака)

Доступ: только чтение

Механизм: по ссылке

Время, в которое срабатывает таймер. Аргумент DAYTIM - это адрес квадрослова, содержащего значение времени. Положительное значение времени задает абсолютное время срабатывания таймера; отрицательное значение времени задает смещение (дельта-время) от текущего времени.

Если заданное значение абсолютного времени уже прошло, то таймер срабатывает в следующем цикле часов, который наступает в пределах 10 миллисекунд.

Программа системного обслуживания "преобразовать строку в коде КОИ-8 в двоичное время" (¤BINTIM) преобразует

значение времени, представленное в коде КОИ-8, в значение времени размером в квадрослово, необходимое $SETIMR.

ASTADR

Использование в МОС ВП: AST_PROCEDURE

Тип: маска входа программы

Доступ: вызов без развертывания стека

Механизм: по ссылке

Подпрограмма обработки AST, которая должна выполняться, когда сработает таймер. Аргумент ASTADR - это адрес маски входа данной подпрограммы. Если ASTADR не задан или равен 0 (по умолчанию), то подпрограмма AST не выполняется.

Подпрограмма AST, если она указана, работает в режиме доступа вызывающей программы.

REQIDT

Использование в МОС ВП: USER_ARG

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Идентификатор запроса к таймеру. Аргумент REQIDT - это значение длинного слова, содержащего номер, который однозначно идентифицирует данный запрос к таймеру. Если REQIDT не задан, то используется значение 0.

Для того, чтобы отменить запрос к таймеру, программе служебного обслуживания "отменить таймер" ($CANTIM) передается (как аргумент REQIDT) идентификатор данного запроса.

Если требуется отменить не все, а конкретные запросы к таймеру, нужно обеспечить, чтобы в вызове $SETIMR для аргу-

мента REQIDT было указанно ненулевое значение, потому что ¤CANIMR интерпретирует нуль, заданный в качестве значения идентификатора, как запрос на отмену всех запросов к таймеру.

Для каждого запроса к таймеру можно задать уникальные значения, но взаимосвязанным запросам к таймеру можно задать одинаковые значения. Это позволяет выборочно отменять один запрос к таймеру, группу взаимосвязанных запросов или все запросы к таймеру.

Если в вызове ¤SETIMR задан аргумент ASTADR, то значение, заданное аргументом REQIDT, передается как параметр подпрограмме AST. Если подпрограмма AST требует несколько параметров, то в качестве значения для REQIDT указывается адрес; тогда подпрограмма AST может интерпретировать данный адрес, как начало списка параметров.

Описание.

Программа системного обслуживания "установить таймер" требует динамической памяти и использует квоту входов в очередь к таймеру (TQELM), выделенную процессу. Если задана подпрограмма AST, то программа системного обслуживания использует квоту предела AST (ASTLM), выделенную процессу.

Программа системного обслуживания ¤SETIMR работает в режиме доступа вызывающей программы, так же, как и подпрограмма AST, если она указана.

Возвращаемые значения кода состояния:

SS$_NORMAL - успешное завершение;

- SS$_ACCVIO вызывающая программа не может прочитать время срабатывания;
- SS$_EXQUOTA процесс превысил свою квоту входов в очередь к таймеру или квоту предела AST; или недостаточно системной динамической памяти для выполнения запроса;
- SS$_ILLEFC задан недопустимый номер флага события;
- SS$_INSFMEM не хватает динамической памяти для назначения входа в очередь к таймеру;
- SS$_UNASEFC процесс не связан с кластером, содержащим заданный флаг события.

13.95. $SETPRA - установить AST восстановления питания

Программа системного обслуживания $SETPRA устанавливает подпрограмму, которая получает управление после восстановления питания.

Формат:

SYS$SETPRA ASTADR,[ACMODE]

Аргументы:

ASTADR

Использование в МОС ВП: AST_PROCEDURE

Тип: маска входа в программу

Доступ: вызов без развертывания стека

Механизм: по ссылке

Подпрограмма обработки AST восстановления питания, которая должна получить управление после того, как произойдет восстановление питания. Аргумент ASTADR является адресом маски входа в данную подпрограмму.

Если агрумент ASTADR имеет значение 0, то прерывание AST не доставляется процессу при восстановлении питания.

Операционная система МОС ВП передает указанной подпрограмме AST один параметр. Данный параметр является словом, значение которого содержит промежуток времени, в течении которого было отключено питание, выраженное в виде числа интервалов в 1/100 секунды.

ACMODE

Использование в МОС ВП: ACCESS_MODE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Режим доступа, в котором должна выполняться программа восстановления питания. Аргумент ACMODE является длинным словом, содержащим данный режим доступа. Макрокоманда ¤PCLDEF определяет символические имена для четырех режимов доступа.

Наиболее привилегированным режимом доступа является режим доступа вызывающего процесса.

Описание.

Программа системного обслуживания ¤SETPRA использует квоту лимита AST процесса (ASTLM).

Для процесса можно указать только одну подпрограмму

AST восстановления питания. При выходе образа адрес входа в подпрограмму AST очищается.

Соглашения о входе и выходе для подпрограммы AST восстановления питания такие же как и для всех подпрограмм обработки AST. Данные соглашения описаны в разделе 5.

Возвращаемые значения кода состояния:

SS¤_NORMAL - программа системного обслуживания успешно завершилась;

- SS¤_EXQUOTA процесс превысил свою квоту для необработанных запросов AST.

13.96. ¤SETPRI - установить приоритет

Программа системного обслуживания ¤SETPRI изменяет базовый приоритет процесса. Базовый приоритет определяет, в каком порядке запускаются выполняемые процессы.

Формат:

SYS¤SETPRI [PIDADR],[PRCNAM],PRI,[PRVPRI]

Аргументы:

PIDADR

Использование в МОС ВП: PROCESS_ID

Тип: длинное слово (без знака)

Доступ: модификация

Механизм: по ссылке

Идентификатор процесса (PID) того процесса, чей приоритет устанавливается в результате работы данной программы. Аргумент PIDADR является адресом идентификатора PID.

PRCNAM

Использование в МОС ВП: PROCESS_NAME

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Имя процесса, приоритет которого должен быть изменен в результате работы данной программы. Аргумент PRCNAM является адресом дескриптора строки символов, указывающего на строку имени процесса длиной от 1 до 15 символов.

Аргумент можно использовать только для процессов, принадлежащих той же группе UIC, что и вызывающий процесс. Чтобы установить приоритет для процессов, принадлежащих другим группам, следует задавать аргумент PIDADR.

Если не задан ни один из аргументов PIDADR и PRCNAM, то программа $SETPRI устанавливает базовый приоритет вызывающего процесса.

PRI

Использование в МОС ВП: LONGWORD_USINGNED

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Новый базовый приоритет, который должен быть установлен для процесса. Аргумент PRI является длинным словом, значение которого содержит новый приоритет. Приоритеты, не относящиеся к режиму реального времени, лежат в диапазоне от 0 до 15, а приоритеты, относящиеся к режиму реального

времени лежат в диапазоне от 16 до 31.

Если заданный приоритет выше базового приоритета соответствующего процесса, а вызывающий процесс не имеет привилегии ALTPRI, то используется базовый приоритет соответствующего процесса.

PRVPRI

Использование в МОС ВП: LONGWORD_USINGNED

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Базовый приоритет процесса перед вызовом программы $SETPRI. Аргумент PRVPRI является адресом длинного слова, в которое программа $SETPRI записывает предыдущий приоритет процесса.

Описание.

В зависимости от операции, использование программы $SETPRI может потребовать, чтобы вызывающий процесс имел одну из следующих привилегий:

1) чтобы изменить приоритет процесса в той же группе, требуется привилегия GROUP, если только процесс, приоритет которого изменяется, не имеет тот же идентификатор UIC, что и вызывающий процесс;

2) чтобы изменить приоритет любого другого процесса в системе, требуется привилегия WORLD;

3) чтобы изменить базовый приоритет процесса на большее значение, требуется привилегия ALTPRI.

Базовый приоритет процесса действует до тех пор, пока

он не будет специально изменен, или пока процесс не будет удален.

Если процесс не имеет привилегии ALTPRI и пытается установить другому процессу приоритет, значение которого выше базового приоритета данного другого процесса, то ему устанавливается значение приоритета равное базовому приоритету и возвращается код состояния SS$_NORMAL.

Чтобы определить приоритет, установленный программой системного обслуживания $SETPRI, следует использовать программу системного обслуживания "получить информацию о задании или процессе" ($GETJPI).

Возвращаемые значения кода состояния:

SS$_NORMAL - программа системного обслуживания успешно завершилась;

- SS$_ACCVIO вызывающий процесс не может получить строку имени процесса или дескриптор строки; или не может записать идентификатор процесса или длинное слово предыдущего приоритета;
- SS$_IVLOGNAM строка имени процесса имеет нулевую длину или длину большую 15 символов;
- SS$_NONEXPR предупреждение. Указанный процесс не существует или указан неверный идентификатор процесса;
- SS$_NOPRIV процесс не имеет привилегий для оказания влияния на другие процессы.

13.97. ¤SETPRN - установить имя процесса

Программа системного обслуживания ¤SETPRN позволяет процессу создать или изменить свое собственное имя процесса.

Формат:

SYS¤SETPRN [PRCNAM]

Аргументы:

PRCNAM

Использование в МОС ВП: PROCESS_NAME

Тип: строка текста

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки с фиксированной длиной

Имя, которое надо дать вызывающему процессу. Аргумент PRCNAM - это адрес дескриптора строки символов, указывающего на строку имени процесса длиной от 1 до 15 символов. Если аргумент PRCNAM не задан, вызывающий процесс не получает имени.

Описание.

Имя процесса остается в действии до тех пор, пока оно не изменится (при помощи ¤SETPRN), либо пока процесс не будет удален.

Имена процесса обеспечивают механизм идентификации для процессов, работающих с одинаковым номером группы. Процесс можно также идентифицировать при помощи его идентификатора (PID).

Возвращаемые значения кода состояния:

SS$_NORMAL - успешное завершение;

- SS$_ACCVIO вызывающая программа не может прочитать строку имени процесса или дескриптор строки;

- SS$_DUPLNAM заданное имя процесса дублирует уже существующее имя в данной группе UIC;

- SS$_IVLOGNAM заданное имя процесса имеет нулевую длину или длину более 15 символов.

13.98. $SETPRT - установить защиту страниц

Программа системного обслуживания $SETPRT позволяет процессу изменить защиту одной страницы или диапазона страниц.

Формат:

SYS$SETPRT INADR,[RETADR],[ACMODE],PROT,[PRVPRT]

Аргументы:

INADR

Использование в МОС ВП: ADDRESS_RANGE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Начальный и конечный виртуальные адреса диапазона страниц, для которых надо изменить защиту. Аргумент INADR - это адрес массива из двух длинных слов, содержащих по порядку начальный и конечный виртуальные адреса процесса. В каждом виртуальном адресе используется только та часть,

которая представляет номер виртуальной страницы, младшие 9 битов игнорируются.

Если начальный виртуальный адрес равен конечному, то изменяется защита для одной страницы.

RETADR

Использование в МОС ВП: ADDRESS_RANGE

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Начальный и конечный виртуальные адреса диапазона страниц, для которых $SETPRT фактически изменила защиту. Аргумент RETADR - это адрес массива из двух длинных слов, содержащих по порядку начальный и конечный виртуальные адреса процесса.

Если во время изменения защиты возникла ошибка, то $SETPRT записывает в RETADR диапазон страниц, которые были успешно изменены до возникновения ошибки. Если до возникновения ошибки таких страниц не было, то $SETPRT записывает в каждое длинное слово массива значение "минус 1".

ACMODE

Использование в МОС ВП: ACCESS_MODE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Режим доступа, связанный с $SETPRT. Аргумент ACMODE - это длинное слово, содержащее режим доступа. Макрокоманда $PSLDEF определяет обозначения для четырех режимов доступа.

¤SETPRT использует наименее привилегированный из следующих режимов доступа: режим доступа, заданный аргументом ACMODE и режим доступа вызывающего процесса. Для того, чтобы можно было изменить защиту любой страницы в заданном диапазоне, результирующий режим доступа должен иметь равную или высшую привилегию по сравнению с режимом доступа владельца данной страницы.

PROT

Использование в МОС ВП: PAGE_PROTECTION

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Защита страниц, которую надо присвоить указанным страницам. Аргумент PROT - это значение длинного слова, содержащего код защиты. Используются только биты с 0 по 3; биты с 4 по 31 игнорируются.

Макрокоманда ¤PRTDEF определяет следующие символические имена для кодов защиты:

1) PRT¤C_NA - нет доступа;

2) PRT¤C_KR - только чтение в режиме ядра;

3) PRT¤C_KW - запись в режиме ядра;

4) PRT¤C_ER - только чтение в режиме управления;

5) PRT¤C_EW - запись в режиме управления;

6) PRT¤C_SR - только чтение в режиме супервизора;

7) PRT¤C_SW - запись в режиме супервизора;

8) PRT¤C_UR - только чтение в режиме пользователя;

9) PRT¤C_UW - запись в режиме пользователя;

10) PRT¤C_ERKW - чтение в режиме управления, запись в режиме ядра;

11) PRT¤C_SRKW - чтение в режиме супервизора, запись в режиме ядра;

12) PRT¤C_SREW - чтение в режиме супервизора, запись в режиме управления;

13) PRT¤C_URKW - чтение в режиме пользователя, запись в режиме ядра;

14) PRT¤C_UREW - чтение в режиме пользователя, запись в режиме управления;

15) PRT¤C_URSW - чтение в режиме пользователя, запись в режиме супервизора.

Если защита задана как 0, то по умолчанию принимается только чтение в режиме ядра.

PRVPRT

Использование в МОС ВП: PAGE_PROTECTION

Тип: байт (без знака)

Доступ: только запись

Механизм: по ссылке

Защита, присвоенная раньше последней странице в данном диапазоне. Аргумент PRVPRT - это адрес байта, в который ¤SETPRT записывает защиту данной страницы. Аргумент PRVPRT полезен только в том случае, когда изменяется защита одной страницы.

Описание:

Если процесс изменяет защиту каких-либо страниц в личной секции с доступа "только чтение" на доступ "чтение/запись", то $SETPRT использует квоту файла обмена страниц (PGFLQUOTA) процесса.

Для страниц в глобальных секциях можно вводить новую защиту только для страниц, которые являются копируемыми по ссылке.

Возвращаемые значения кода состояния:

- SS$_NORMAL успешное завершение;
- SS$_ACCVIO вызывающая программа не может прочитать массив входных адресов; либо не может записать массив выходных адресов или байт, предназначенный для получения прежней защиты; либо была сделана попытка изменить защиту не существующей страницы;
- SS$_EXQUOTA процесс превысил свою квоту файла обмена страниц во время изменения защиты страницы в личной секции доступа "только чтение" на доступ "чтение/запись";
- SS$_LENVIO страница в заданном диапазоне находится за концом области программы или области управления;
- SS$_NOPRIV страница в заданном диапазоне находится в системном адресном пространстве;
- SS$_PAGOWNVI нарушения владения страницей. Была сде-

лана попытка изменить защиту страницы, которая принадлежит более привилегированному режиму доступа.

13.99. ¤SETPRV - установить привилегии

Программа системного обслуживания ¤SETPRV разрешает или запрещает для вызывающего процесса заданные привилегии.

Формат:

SYS¤SETPRV [ENBFLG],[PRVADR],[PRMFLG],[PRVPRV]

Аргументы:

ENBFLG

использование в MOC ВП: BOOLEAN

тип: байт (без знака)

доступ: только чтение

механизм: по значению

Индикатор, который указывает, необходимо разрешить или запретить заданные привилегии. Аргумент ENBFLG - это однобайтовое значение. Значение 1 указывает, что привилегии, заданные аргументом PRVADR, необходимо разрешить. Значение 0 (принимаемое по умолчанию) указывает, что привилегии необходимо запретить.

PRVADR

использование в MOC ВП: MASK_PRIVILEGES

тип: квадрослово (без знака)

доступ: только чтение

механизм: по ссылке

Привилегии, которые необходимо разрешить или запретить для вызывающего процесса. Аргумент PRVADR - это адрес битового вектора размером в квадрослово, в котором каждый бит соответствует некоторой привилегии, запрещаемой или разрешаемой.

Каждый бит имеет символическое имя; данные имена определяет макрокоманда $PRVDEF. Битовый вектор формируется при помощи задания символического имени каждой требуемой привилегии в операции "логическое или". В табл. 68 приведены символические имена и описания привилегий.

Таблица 68

Привилегии

Привилегии	Символическое имя	Допустимые действия
ALLSPOOL	PRV$V_ALLSPOOL	назначить устройство, обслуживающее через спул
BUGCHK	PRV$V_BUGCHK	создавать входы в журнал ошибок
BYPASS	PRV$V_BYPASS	обходить защиту, основанную на UIC
CMEXEC	PRV$V_CMEXEC	изменить режим на режим управления
CMKRNL	PRV$V_CMKRNL	изменить режим на режим ядра
DETACH	PRV$V_DETACH	создавать отсоединенные процессы

Продолжение табл. 68

Привилегии	Символическое имя	Допустимые действия
DIAGNOSE	PRV$V_DIAGNOSE	осуществлять диагностику устройств
DOWNGRADE	PRV$V_DOWNGRADE	понижать классификацию
EXQUOTA	PRV$V_EXQUOTA	превышать квоты
GROUP	PRV$V_GROUP	управлять группой процессов
GRPNAM	PRV$V_GRPNAM	помещать имя в групповую таблицу логических имен
GRPPRV	PRV$V_GRPPRV	осуществлять доступ в группе через поля с системной защитой
LOG_IO	PRV$V_LOG_IO	выполнять операции логического ввода-вывода
MOUNT	PRV$V_MOUNT	ставить в очередь ввода-вывода запрос на монтирование тома
NETMBX	PRV$V_NETMBX	создавать устройство сети
ACNT	PRV$V_NOACNT	создавать процессы, для которых не ведется учет
OPER	PRV$V_OPER	все операторские привилегии
PFNMAP	PRV$V_PFNMAP	отображать секцию по номеру физической страницы
PHY_IO	PRV$V_PHY_IO	выполнить операции физического ввода-вывода
PRMCEB	PRV$V_PRMCEB	создавать постоянные кластеры

Продолжение тэбл. 68

Привилегии	Символическое имя	Допустимые действия
		общих флагов событий
PRMGBL	PRV$V_PRMGBL	создавать постоянные глобальные секции
PRMJNL	PRV$V_PRMJNL	создавать постоянные журналы
PRMMBX	PRV$V_PRMMBX	создавать постоянные почтовые ящики
PSWAPM	PRV$V_PSWAPM	изменять режим обмена процессов
READALL	PRV$V_READALL	иметь доступ на чтение ко всем данным
SECURITY	PRV$V_SECURITY	выполнять операции по обеспечению защиты
ALTPRI	PRV$V_SETPRI	устанавливать (изменять) любой приоритет процесса
SETPRV	PRV$V_SETPRV	устанавливать любые привилегии процесса
SHARE	PRV$V_SHARE	назначать канал неразделяемому устройству
SHMEM	PRV$V_SHMEM	создавать структуры в памяти, разделяемой несколькими процессорами
SYSGBL	PRV$V_SYSGBL	создавать системные глобальные

Продолжение табл. 68

Привилегии	Символическое имя	Допустимые действия
		секции
SYSLCK	PRV$V_SYSLCK	ставить в очередь захват на общесистемные ресурсы
SYSNAM	PRV$V_SYSNAM	помещать имя в системную таблицу логических имен
SYSPRV	PRV$V_SYSPRV	получать доступ к файлам и другим процессам так, как если бы процесс имел системный UIC
TMPJNL	PRV$V_TMPJNL	создавать временные журналы
TMPMBX	PRV$V_TMPMBX	создавать временные почтовые ящики
UPGRADE	PRV$V_UPGRADE	повышать классификацию
VOLPRO	PRV$V_VOLPRO	подавлять защиту тома
WORLD	PRV$V_WORLD	управлять всеми процессами

Имена битов привилегий PRV$V_NOACNT и PRV$V_SETPRI соответствуют именам привилегий диалогового командного языка ACNT и ALTPRI, хотя они и различаются по написанию.

Если аргумент PRVADR не указан или равен 0, то привилегии не изменяются.

PRMFLG

Использование в МОС ВП: BOOLEAN

Тип: байт (без знака)

Доступ: только чтение

Механизм: по значению

Индикатор, который указывает, должны привилегии изменяться временно или постоянно. Аргумент PRMFLG - это однобайтовое значение. Значение 1 указывает, что привилегии необходимо изменить навсегда, то есть, до тех пор, пока их снова не изменят при помощи $SETPRV или пока процесс не будет удален. Значение 0 (принимаемое по умолчанию) указывает, что привилегии необходимо изменить временно, то есть, до тех пор, пока текущий образ не осуществит выход (в то же время постоянно разрешаемые привилегии процесса будут сохранены).

PRVPRV

Использование в МОС ВП: MASK_PRIVILEGES

Тип: квадрослово (без знака)

Доступ: только запись

Механизм: по ссылке

Привилегии, которыми прежде обладал вызывающий процесс. Аргумент PRVPRV - это адрес битового вектора размером в квадрослово, в котором каждый бит соответствует некоторой привилегии, разрешенной или запрещенной ранее. Если аргумент PRVPRV не задан или равен 0, то маска прежних привилегий не возвращается.

Описание.

Чтобы установить какую-либо привилегию постоянно, вызывающий процесс должен быть авторизован на установку данной привилегии, либо процесс должен выполняться в режиме ядра или в режиме управления.

Для временной установки какой-либо привилегии нужно, чтобы выполнялось одно из следующих условий:

1) вызывающий процесс должен быть авторизован на установку заданной привилегии;

2) вызывающий процесс должен выполняться в режиме ядра или в режиме управления.

3) текущий работающий образ должен быть образом, который был установлен с заданной привилегией.

Операционная система МОС ВП поддреживает четыре отдельных маски привилегий для каждого процесса:

AUTHPRIV -

Привилегии, которые авторизованно разрешены процессу так, как их назначил администратор системы или создатель процесса. Маска AUTHPRIV никогда не изменяется в течение жизни процесса;

PROCPRIV -

Привилегии, которые назначаются процессу, как временно разрешенные. Маску PROCPRIV можно модифицировать при помощи $SETPRV;

IMAGPRIV -

Привилегии, с которыми был установлен текущий образ;

CURPRIV -

Привилегии, которые разрешены на текущий момент. Маску CURPRIV можно модифицировать при помощи $SETPRV.

При создании процесса его маски AUTHPRIV, PROCPRIV и CURPIV имеют одинаковое содержание. Всякий раз, когда какой-либо программой системного обслуживания (отличной от $SETPRV) нужно проверить привилегии процесса, данная программа проверяет маску CURPRIV.

Когда процесс выполняет установленный образ, привилегии, с которыми был установлен данный образ, включаются в маску CURPRIV. Когда установленный образ осуществляет выход, в маску CURPRIV копируется маска PRCPRIV.

Программа системного обслуживания $SETPRV может устанавливать биты только в масках CUPRIV и PRCPRIV, но $SETPRV проверяет маску AUTHPRIV, когда требуется узнать, может ли процесс устанавливать заданные биты привилегий в масках CURPIV или PROCPRIV. Следовательно, процесс может дать себе привилегию SETPRV только в том случае, если данная привилегия разрешена в маске AUTHPRIV.

Каждую из четырех масок привилегий процесса можно получить при помощи вызова программы системного обслуживания "получить информацию о задании или процессе" ($GETJPI), указав требуемую маску (маски) привилегий как код (коды) элемента в аргументе ITMLST. Код элемента для маски привилегии строится при помощи добавления символического префикса JPI$_ к имени маски привилегии (например, JPI$_CURPRIV - это код элемента для текущей маски привилегий).

Команда диалогового командного языка SET PROCESS/PRIVILEGES также разрешает или запрещает заданные привилегии.

Возвращаемые значения кода состояния:

SS$_NORMAL - успешное завершение. Все привилегии были разрешены или запрещены, как указано;

SS$_NOTALLPRIV - успешное завершение; не все указанные привилегии были разрешены;

SS$_ACCVIO - вызывающая программа не может прочитать маску привилегий или не может записать прежнюю маску привилегий.

13.100. $SETRWM - установить режим ожидания ресурса

Программа системного обслуживания $SETRWM позволяет процессу указать, какие действия должны предпринимать программы системного обслуживания, когда необходимые для их выполнения системные ресурсы недоступны.

Если режим ожидания ресурса разрешен, программы системного обслуживания будут ждать, когда требуемые системные ресурсы станут доступными, а затем продолжат выполнение.

Если режим ожидания ресурса запрещен, программы системного обслуживания будут возвращать управление вызывающей программе, если требуемые системные ресурсы недоступны.

Код состояния, возвращаемый программой $SETRWM, показывает, был ли режим ожидания ресурса разрешен или запре-

щен.

Формат:

SYS¤SETRWM [WATFLG]

Аргументы:

WATFLG

Использование в МОС ВП: LONGWORD_UNSIGNED

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Индикатор, показывающий, должны ли программы системного обслуживания ожидать требуемые ресурсы. Аргумент WATFLG - это значение размером в длинное слово. Значение 0 (принимаемое по умолчанию) указывает, что программы системного обслуживания должны ждать, пока ресурсы, необходимые для их выполнения, не станут доступными.

Значение 1 указывает, что программы системного обслуживания, должны сразу же возвращать управление с кодом состояния ошибки, если необходимые для их выполнения ресурсы недоступны.

Операционная система МОС ВП разрешает режим ожидания ресурса для всех процессов. Режим ожидания ресурса можно запретить только при помощи вызова ¤SETRWM. Если режим ожидания ресуса запрещен, то он остается запрещенным до тех пор, пока его явно не разрешат, или пока процесс не будет удален.

Описание.

Режим ожидания ресурса использует следующие системные ресурсы и квоты процесса:

1) системную динамическую память;

2) регистры отображения адаптера общей шины;

3) квоту предела прямого ввода-вывода (DIOLM);

4) квоту предела буферизованного ввода-вывода (BIOLM);

5) квоту предела счетчика байтов буферизованного ввода-вывода (BYTLM).

Возвращаемые значения кода состояния:

SS¤_WASCLR - успешное завершение. Режим ожидания ресурса был прежде разрешен;

SS¤_WASSET - успешное завершение. Режим ожидания ресурса был прежде запрещен.

13.101. ¤SETSFM - установить режим исключительной ситуации по ошибке программы системного обслуживания

Программа системного обслуживания ¤SETSFM позволяет процессу указать, необходимо или нет, чтобы операционная система MDC ВП создавала исключительную ситуацию, порожденную программным обеспечением, в тех случаях, когда программа системного обслуживания возвращает вызывающему процессу код состояния ошибки.

¤SETSFM указывает в значении кода возврата, был режим исключительной ситуации по ошибке программы системного обслуживания разрешен или запрещен прежде (до вызова ¤SETSFM).

Режим исключительной ситуации по ошибке программы системного обслуживания изначально запрещен, поэтому вызывающему процессу следует после вызова программы системного обслуживания явно проверять, насколько успешным было ее завершение.

Формат:

SYS$SETSFM [ENBFLG]

Аргументы:

ENBFLG

Использование в MOC ВП: BOOLEAN

Тип: байт (без знака)

Доступ: только чтение

Механизм: по значению

Число, определяющее, разрешается ли режим исключительной ситуации по ошибке программы системного обслуживания. Аргумент ENBFLG - это однобайтовое значение. Значение 1 указывает, что режим исключительной ситуации по ошибке программы системного обслуживания разрешается. Значение 0 (принимаемое по умолчанию) указывает, что режим исключительной ситуации по ошибке программы системного обслуживания запрещается.

Описание:

Если данный режим разрешен, то, когда программа системного обслуживания возвращает код состояния ошибки, создается исключительная ситуация, вызванная программным обеспечением. Исключительные ситуации по ошибке программы системного обслуживания создаются только в том случае, если

вызов программы системного обслуживания происходит от пользовательского режима. Однако, программу системного обслуживания ¤SETSFM можно вызывать из любого режима доступа.

Режим исключительной ситуации по ошибке программы системного обслуживания, если он разрешен, остается разрешенным до тех пор, пока его явно не запретят или пока образ не осуществит выход. Можно указать обработчик кода состояния в первом длинном слове стека вызовов программы при помощи программы системного обсуживания "установить вектор исключительной ситуации" (¤SETEXV). Если пользователь не указал обработчик условия, то используется принимаемый по умолчанию системный обработчик. Данный обработчик кода состояния организует выход образа и затем выводит на дисплей состояние выхода.

Список аргументов, который передается обработчику кода состояния, содержит в аргументе "имя состояния" сигнального массива код SS¤_SSFAIL.

Возвращаемые значения кода состояния:

SS¤_WASCLR - успешное завершение. Исключительные ситуации по ошибке были прежде запрещены;

SS¤_WASSET - успешное завершение. Исключительные ситуации по ошибке были прежде разрешены.

13.102. ¤SETSSF - установить фильтр программ системного обслуживания

Программа системного обслуживания ¤SETSSF подавляет вызовы из пользовательского режима определенных программ

системного обслуживания. При помощи программы ¤SETSSF нельзя подавить вызовы следующих программ системного обслуживания:

1) ¤ASCTIM;

2) ¤BINTIM;

3) ¤EXIT;

4) ¤FAO;

5) ¤FAOL;

6) ¤PUTMSG;

7) ¤UNWIND.

Формат:

SYS¤SETSSF [MASK]

Аргументы:

MASK

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Категория программ системного обслуживания, которые нужно подавлять для вызовов из пользовательского режима. Аргумент MASK - это значение длинного слова, в котором значащим является только первый байт. Первый байт - это битовый вектор, в котором установленный бит задает категорию программ системного обслуживания, которые нужно подавить. Используются только биты 0 и 1; биты со 2 по 7 игнорируются.

Если установлен бит 0, то подавляются все программы

системного обслуживания, включая написанные пользователем программы системного обслуживания.

Если установлен бит 1, то подавляются все программы системного обслуживания, включая написанные пользователем программы системного обслуживания, кроме следующих:

1) ¤ADJSTK;

2) ¤CRETVA;

3) ¤DELTVA;

4) ¤GETMSG;

5) ¤SETSFM.

Бит 1 подавляет меньшее количество программ системого обслуживания, чем бит 0. Конкретно, бит 1 не подавляет программы системного обслуживания, которые необходимы для программ системного обслуживания обработки кода состояния и свертывания образа, тогда как бит 0 это делает.

Описание:

Для успешного вызова ¤SETSSF надо, чтобы режим доступа вызывающей программы имел равную или высшую привилегию по сравнению с супервизорным режимом доступа, и чтобы при загрузке системы был установлен параметр системной генерации (SYSGEN) SSINHIBIT.

Если программа системного обслуживания, которая уже подавлена, вызывается из пользовательского режима, то возвращается одно из следующих двух значений кода состояния:

SS¤_INHCHME - вызвана запрещенная программа системного обслуживания с режимом управления;

SS¤_INHCHMK - вызвана запрещенная программа системного

обслуживания с режимом ядра.

Возвращаемые значения кода состояния:

SS$_NORMAL - успешное завершение;

SS$_NOPRIV - процесс не имеет привилегии на вызов данной программы системного обслуживания.

13.103. $SETSTK - установить границы стека

Программа системного обслуживания $SETSTK позволяет процессу изменить размер его стеков режима супервизора, режима управления и режима ядра посредством изменения значений в массивах размера и основания стека, содержащихся в области P1 (для каждого процесса).

Формат:

SYS$SETSTK INADR,[RETADR],[ACMODE]

Аргументы:

INADR

Использование в МОС ВП: ADDRESS_RANGE

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Диапазон адресов, который представляет новые границы стека. Аргумент INADR - это адрес массива из двух длинных слов, содержащих адрес верхушки стека и адрес основания стека. Поскольку стеки в области P1 расширяются по направлению от старших адресов к младшим, адрес основания должен быть больше, чем адрес верхушки стека.

RETADR

Использование в МОС ВП: ADDRESS_RANGE

Тип: длинное слово (без знака)

Доступ: только запись

Механизм: по ссылке

Диапазон адресов, который представляет прежние границы стека. Аргумент RETADR - это адрес массива из двух длинных слов, в которые $SETSTK записывает, в первое длинное слово - прежний адрес верхушки стека, и во второе длинное слово- прежний адрес основания стека.

ACMODE

Использование в МОС ВП: ACCESS_MODE

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Режим доступа изменяемого стека. Аргумент ACMODE - это длинное слово, содержащее режим доступа. Макрокоманда $PSLDEF определяет обозначения для четырех режимов доступа. Наиболее привилегированный режим доступа, который может использоваться - это режим доступа вызывающего процесса.

Если аргумент ACMODE задает пользовательский режим, то $SETSTK не выполняет никаких действий и возвращает код состояния SS$_NORMAL.

Описание:

Вызывающий процесс может регулировать размеры стеков только для тех режимов доступа, которые имеют равную или меньшую привилегию по сравнению с режимом доступа вызываю-

щего процесса.

Возвращаемые значения кода состояния:

SS¤_NORMAL - успешное завершение;

SS¤_ACCVIO - вызывающая программа не может прочитать массив входных адресов; входной диапазон неверен; или вызывающая программа не может записать массив возвращаемых адресов.

13.104. ¤SETSWM - установить режим обмена процесса

Программа системного обслуживания ¤SETSWM позволяет процессу указать, может или нет он выгружаться из балансного набора.

Когда режим обмена процесса разрешен, процесс может быть выгружен; когда он запрещен, процесс остается в балансном наборе до тех пор, пока снова не будет разрешен режим обмена, или пока процесс не будет удален.

Программа системного обслуживания ¤SETSWM возвращает значение кода состояния, которое показывает, был режим обмена процесса разрешен или запрещен до вызова ¤SETSWM.

Для того, чтобы зафиксировать некоторые, но необязательно все, страницы процесса в балансном наборе, следует использовать программу системного обслуживания "зафиксировать страницы в памяти" (¤LCKPAG).

Формат:

SYS$SETSWM [SWPFLG]

Аргументы:

SWPFLG

Использование в МОС ВП: LONGWORD_UNSIGNED

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Индикатор, который показывает, можно выгружать процесс или нет. Аргумент SWPFLG - это значение размером в длинное слово. Значение 0 (принимаемое по умолчанию) разрешает режим обмена процесса, то есть процесс может быть выгружен. Значение 1 запрещает режим обмена процесса, то есть процесс нельзя выгрузить.

Описание:

Чтобы изменить свой режим обмена, вызывающий процесс должен иметь привилегию $PSWAPM.

Возвращаемые значения кода состояния:

SS$_WASCLR - успешное завершение. Процесс прежде не был зафиксирован в балансном наборе;

SS$_WASSET - успешное завершение. Процесс прежде был зафиксирован в балансном наборе;

SS$_NOPRIV - процесс не имеет необходимой привилегии PSWAPM.

13.105. ¤SNDERR - послать сообщение в журнал регистрации ошибок

Программа системного обслуживания ¤SNDERR записывает в системный файл регистрации ошибок сообщение, сформированное пользователем, с указанием даты и времени в начале сообщения.

Формат:

SYS¤SNDERR MSGBUF

Аргументы:

MSGBUF

Использование в МОС ВП: SHAR_STRING

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Сообщение, которое записывается в файл регистрации ошибок. Аргумент MSGBUF является адресом дескриптора строки символов, указывающего на текст сообщения.

Описание.

Чтобы отослать сообщение в файл регистрации ошибок, вызывающий процесс должен иметь привилегию BUGCHK.

Программе системного обслуживания ¤SNDERR требуется системная динамическая память.

Возвращаемые значения кода состояния:

SS¤_NORMAL - программа системного обслуживания успешно завершилась;

SS¤_ACCVIO - вызывающий процесс не может прочитать буфер сообщения или дескриптор буфера;

SS¤_INSFMEM - для завершения программы системного обслуживания нет доступной системной динамической памяти достаточного размера;

SS¤_NOPRIV - процесс не имеет требуемой привилегии BUGCHK.

13.106. ¤SNDJBC - сообщить диспетчеру заданий

Программа системного обслуживания ¤SNDJBC создает и приостанавливает очереди, а также управляет очередями и пакетными и печатающими заданиями в данных очередях. Программы системного обслуживания ¤SNDJBC и ¤GETQUI в совокупности обеспечивают пользователю интерфейс с диспетчером заданий операционной системы МОС ВП.

Программа системного обслуживания ¤SNDJBC завершается асинхронно, то есть, она возвращает управление вызывающему процессу после постановки в очередь, не ожидая завершения операции.

Для синхронизации завершения большинства операций следует использовать программу системного обслуживания "сообщить диспетчеру заданий и ждать" (¤SNDJBCW). Программа системного обслуживания ¤SNDJBCW во всем идентична ¤SNDJBC кроме того, что ¤SNDJBCW возвращает управление вызывающему

процессу после завершения операции (см. подраздел 2.5).

Формат:

SYS$SNDJBC [EFN],FUNC,[NULLARG],[ITMLST]
,[IOSB],ASTADR],[ASTPRM]

Аргументы:

EFN

Использование в МОС ВП: EF_NUMBER

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Номер того флага события, который нужно установить по завершении $SNDJBC. Аргумент EFN - это длинное слово, содержащее данный номер. Аргумент EFN является необязательным.

Когда запрос ставится в очередь, $SNDJBC очищает указанный флаг события (или флаг события 0, если аргумент EFN не был задан). Потом, когда операция завершится, $SNDJBC устанавливает в 1 указанный флаг события (или флаг события 0).

FUNC

Использование в МОС ВП: FUNCTION_CODE

Тип: слово (без знака)

Доступ: только чтение

Механизм: по значению

Код функции, который указывает, какую функцию должна выполнить программа $SNDJBC. Аргумент FUNC - это слово, содержащее данный код функции. Макрокоманда $SJCDEF опреде-

ляет имена для каждого кода функции.

В одном вызове $SNDJBC можно задавать один код функции. Большинство кодов функции требуют или допускают передачу дополнительной информации вызывающей программе. Данная информация передается при помощи аргумента ITMLST, который задает список из одного или нескольких дескрипторов элемента. В свою очередь, каждый дескриптор элемента задает код элемента, который модифицирует, ограничивает или как-то иначе изменяет действия, обозначенные данным кодом функции.

Коды функций $SNDJBC и допустимые для них коды элементов:

SJC$_ABORT_JOB - данный запрос прерывает выполнение текущего задания. Задание может быть удалено, либо, если оно является повторно выполняемым, может быть снова поставлено в очередь.

Обязательный входной код элемента:

SJC$_QUEUE.

Для пакетных заданий обязательный входной код элемента:

SJC$_ENTRY_NUMBER.

Необязательные входные коды элемента:

1) SJC$_DESTINATION_QUEUE;

2) SJC$_HOLD;

3) SJC$_PRIORITY;

4) SJC$_REQUEUE;

5) SJC$_NO_HOLD.

SJC$_ADD_FILE - данный запрос добавляет файл к откры-

тому заданию, принадлежащему запрашивающему процессу. Данная операция используется как часть последовательности вызовов программы системного обслуживания ¤SNDJBC, создающих задание с одним или несколькими файлами. Первый вызов в данной последовательности задает операцию SJC¤_CREATE_JOB для создания открытого задания. Каждый последующий запрос SJC¤_ADD_FILE связывает с данным заданием дополнительный файл. Чтобы завершить спецификацию пакетного или печатающего задания, делается запрос SJC¤_CLOSE_FILE. Для того, чтобы создать задание, которое содержит только один файл, можно сделать один вызов ¤SNDJBC с указанием кода фукнции SJC¤_ENTER_FILE.

Обязателен один из следующих кодов элемента:

1) SJC¤_FILE_IDENTIFICATION;

2) SJC¤_FILE_SPECIFICATION.

Необязательные входные коды элемента:

1) SJC¤_DELETE_FILE;

2) SJC¤_NO_DELETE_FILE;

3) SJC¤_DOUBLE_SPACE;

4) SJC¤_NO_DOUBLE_SPACE;

5) SJC¤_FILE_BURST;

6) SJC¤_NO_FILE_BURST;

7) SJC¤_FILE_COPIES;

8) SJC¤_FILE_FLAG;

9) SJC¤_NO_FILE_FLAG;

10) SJC¤_FILE_SETUP_MODULES;

11) SJC¤_NO_FILE_SETUP_MODULES;

12) SJC$_FILE_TRAILER;

13) SJC$_NO_FILE_TRAILER;

14) SJC$_FIRST_PAGE;

15) SJC$_NO_FIRST_PAGE;

16) SJC$_LAST_PAGE;

17) SJC$_NO_LAST_PAGE;

18) SJC$_PAGE_HEADER;

19) SJC$_NO_PAGE_HEADER;

20) SJC$_PAGINATE;

21) SJC$_NO_PAGINATE;

22) SJC$_PASSALL;

23) SJC$_NO_PASSALL.

SJC$_ALTER_JOB - данный запрос изменяет параметры существующего задания, которое в настоящий момент не выполняется.

Обязательные входные коды элемента:

1) SJC$_QUEUE;

2) SJC$_ENTRY_NUMBER.

Необязательные входные коды элемента:

1) SJC$_AFTER_TIME;

2) SJC$_NO_AFTER_TIME;

3) SJC$_CHARACTERISTIC_NAME;

4) SJC$_NO_CHARACTERISTICS;

5) SJC$_CHARACTERISTIC_NUMBER;

6) SJC$_NO_CHECKPOINT_DATA;

7) SJC$_CLI;

8) SJC$_NO_CLI;

9) SJC$_CPU_LIMIT;

10) SJC$_NO_CPU_LIMIT;

11) SJC$_NO_DELETE_FILE;

12) SJC$_DESTINATION_QUEUE;

13) SJC$_DOUBLE_SPACE;

14) SJC$_NO_DOUBLE_SPACE;

15) SJC$_FILE_BURST;

16) SJC$_NO_FILE_BURST;

17) SJC$_FILE_COPIES;

18) SJC$_FILE_FLAG;

19) SJC$_NO_FILE_FLAG;

20) SJC$_FILE_SETUP_MODULES;

21) SJC$_NO_FILE_SETUP_MODULES;

22) SJC$_FILE_TRAILER;

23) SJC$_NO_FILE_TRAILER;

24) SJC$_FIRST_PAGE;

25) SJC$_NO_FIRST_PAGE;

26) SJC$_FORM_NAME;

27) SJC$_FORM_NUMBER;

28) SJC$_HOLD;

29) SJC$_NO_HOLD;

30) SJC$_JOB_COPIES;

31) SJC$_JOB_NAME;

32) SJC$_LAST_PAGE;

33) SJC$_NO_LAST_PAGE;

34) SJC$_LOG_DELETE;

35) SJC$_NO_LOG_DELETE;

36) SJC$_LOG_QUEUE;

37) SJC$_LOG_SPECIFICATION;

38) SJC$_NO_LOG_SPECIFICATION;

39) SJC$_LOG_SPOOL;

40) SJC$_NO_LOG_SPOOL;

41) SJC$_LOWERCASE;

42) SJC$_NO_LOWERCASE;

43) SJC$_NOTE;

44) SJC$_NO_NOTE;

45) SJC$_NOTIFY;

46) SJC$_NO_NOTIFY;

47) SJC$_OPERATOR_REQUEST;

48) SJC$_NO_OPERATOR_REQUEST;

49) SJC$_PAGE_HEADER;

50) SJC$_NO_PAGE_HEADER;

51) SJC$_PAGINATE;

52) SJC$_NO_PAGINATE;

53) SJC$_PARAMETR_1 до 8;

54) SJC$_NO_PARAMETERS;

55) SJC$_PASSALL;

56) SJC$_NO_PASSALL;

57) SJC$_PRIORITY;

58) SJC$_RESTART;

59) SJC$_NO_RESTART;

60) SJC$_WSDEFAULT;

61) SJC$_NO_WSDEFAULT;

62) SJC$_WSEXTENT;

63) SJC¤_NO_WSEXTENT;

64) SJC¤_WSQUOTA;

65) SJC¤_NO_WSQUOTA.

SJC¤_ALTER_QUEUE - данный запрос изменяет параметры очереди. На выполнение текущих заданий он не влияет.

Обязательный входной код элемента:

SJC¤_QUEUE

Необязательные входные коды элемента:

1) SJC¤_BASE_PRIORITY;

2) SJC¤_CHARACTERISTIC_NAME;

3) SJC¤_NO_CHARACTERISTICS;

4) SJC¤_CHARACTERISTIC_NUMBER;

5) SJC¤_CPU_DEFAULT;

6) SJC¤_NO_CPU_DEFAULT;

7) SJC¤_CPU_LIMIT;

8) SJC¤_NO_CPU_LIMIT;

9) SJC¤_FILE_BURST;

10) SJC¤_NO_FILE_BURST;

11) SJC¤_FILE_BURST_ONE;

12) SJC¤_FILE_FLAG;

13) SJC¤_NO_FILE_FLAG;

14) SJC¤_FILE_FLAG_ONE;

15) SJC¤_FILE_TRAILER;

16) SJC¤_NO_FILE_TRAILER;

17) SJC¤_FILE_TRAILER_ONE;

18) SJC¤_FORM_NAME;

19) SJC¤_FORM_NUMBER;

```
20) SJC¤_GENERIC_SELECTION;

21) SJC¤_NO_GENERIC_SELECTION;

22) SJC¤_JOB_BURST;

23) SJC¤_NO_JOB_BURST;

24) SJC¤_JOB_FLAG;

25) SJC¤_NO_JOB_FLAG;

26) SJC¤_LIMIT;

27) SJC¤_JOB_RESET_MODULES;

28) SJC¤_NO_JOB_RESET_MODULES;

29) SJC¤_JOB_SIZE_MAXIMUM;

30) SJC¤_NO_JOB_SIZE_MAXIMUM;

31) SJC¤_JOB_SIZE_MINIMUM;

32) SJC¤_NO_JOB_SIZE_MINIMUM;

33) SJC¤_JOB_SIZE_SCHEDULING;

34) SJC¤_NO_JOB_SIZE_SCHEDULING;

35) SJC¤_JOB_TRAILER;

36) SJC¤_NO_JOB_TRAILER;

37) SJC¤_OWNER_UIC;

38) SJC¤_PAGINATE;

39) SJC¤_NO_PAGINATE;

40) SJC¤_PRTECTION;

41) SJC¤_RECORD_BLOCKING;

42) SJC¤_NO_RECORD_BLOCKING;

43) SJC¤_RETAIN_ALL_JOBS;

44) SJC¤_NO_RETAIN_JOBS;

45) SJC¤_RETAIN_ERROR_JOBS;

46) SJC¤_SWAP;
```

47) SJC$_NO_SWAP;

48) SJC$_WSDEFAULT;

49) SJC$_NO_WSDEFAULT;

50) SJC$_WSEXTENT;

51) SJC$_NO_WSEXTENT;

52) SJC$_WSQUOTA;

53) SJC$_NO_WSQUOTA.

SJC$_ASSIGN_QUEUE - данный запрос назначает логическую очередь для очереди выполнения. Логическую очередь задают при помощи кода элемента SJC$_QUEUE; очередь выполнения - при помощи кода элемента SJC$_DESTINATION_QUEUE.

Обязательные коды элемента:

1) SJC$_QUEUE;

2) SJC$_DESTINATION_QUEUE.

SJC$_BATCH_CHECKPOINT - данный запрос организует контрольную точку в пакетном задании. Если запрашивающий процесс не является пакетным процессом, то никакие действия не выполняются.

Обязательный входной код элемента:

SJC$_SHECKPOINT_DATA;

SJC$_CLOSE_DELETE - данный запрос удаляет открытое задание, принадлежащее запрашивающему процессу. Коды элемента не допускаются;

SJC$_CLOSE_JOB - данный запрос завершает спецификацию открытого задания, принадлежащего запрашивающему процессу, и помещает задание в очередь, указанную в том запросе SJC$__CREATE_JOB, который открыл данное задание. Если зап-

рос SJC¤_CLOSE_JOB завершается успешно, то задание больше не является открытым заданием; оно становится обычным пакетным или печатающим заданием.

Необязательный выходной код элемента:

SJC¤_JOB_STATUS_OUTPUT;

SJC¤_CREATE_JOB - данный запрос создает открытое задание для запрашивающего процесса. Если процессу уже принадлежит какое-то открытое задание, то оно удаляется. Открытое задание - это пакетное или печатающее задание, которое еще не полностью специфицировано. После того, как сделан запрос SJC¤_CREATE_JOB, открывающий задание, можно сделать последующие вызовы ¤SNDJBC, используя код функции SJC¤_ADD_FILE для того, чтобы задать файлы, связанные с данным заданием. Наконец, можно завершить спецификацию задания при помощи запроса SJC¤_CLOSE_JOB. Если операция SJC¤_CREATE_JOB завершается успешно, то созданному открытому заданию присваивается номер входа; данному заданию не назначается очередь, указанная в операции SJC¤_CREATE_JOB, до тех пор, пока успешно не завершится SJC¤_CLOSE_JOB.

Обязательный входной код элемента:

SJC¤_QUEUE

Необязательные входные коды элемента:

1) SJC¤_ACCOUNT_NAME;

2) SJC¤_AFTER_TIME;

3) SJC¤_NO_AFTER_TIME;

4) SJC¤_CHARACTERISTIC_NAME;

5) SJC¤_NO_CHARACTERISTICS;

6) SJC$_CHARACTERISTIC_NUMBER;

7) SJC$_CLI;

8) SJC$_NO_CLI;

9) SJC$_CPU_LIMIT;

10) SJC$_NO_CPU_LIMIT;

11) SJC$_FILE_BURST;

12) SJC$_NO_FILE_BURST;

13) SJC$_FILE_BURST_ONE;

14) SJC$_FILE_FLAG;

15) SJC$_NO_FILE_FLAG;

16) SJC$_FILE_FLAG_ONE;

17) SJC$_FILE_TRAILER;

18) SJC$_NO_FILE_TRAILER;

19) SJC$_FILE_TRAILER_ONE;

20) SJC$_FORM_NAME;

21) SJC$_FORM_NUMBER;

22) SJC$_HOLD;

23) SJC$_NO_HOLD;

24) SJC$_JOB_COPIES;

25) SJC$_JOB_NAME;

26) SJC$_LOG_DELETE;

27) SJC$_NO_LOG_DELETE;

28) SJC$_LOG_QUEUE;

29) SJC$_LOG_SPECIFICATION;

30) SJC$_NO_LOG_SPECIFICATION;

31) SJC$_LOG_SPOOL;

32) SJC$_NO_LOG_SPOOL;

33) SJC$_LOWERCASE;

34) SJC$_NO_LOWERCASE;

35) SJC$_NOTE;

36) SJC$_NO_NOTE;

37) SJC$_NOTIFY;

38) SJC$_NO_NOTIFY;

39) SJC$_OPERATOR_REQUEST;

40) SJC$_NO_OPERATOR_REQUEST;

41) SJC$_PARAMETER_1 до 8;

42) SJC$_NO_PARAMETERS;

43) SJC$_PRIORITY;

44) SJC$_RESTART;

45) SJC$_NO_RESTART;

46) SJC$_UIC;

47) SJC$_USERNAME;

48) SJC$_WSDEFAULT;

49) SJC$_NO_WSDEFAULT;

50) SJC$_WSEXTENT;

51) SJC$_NO_WSEXTENT;

52) SJC$_WSQUOTA;

53) SJC$_NO_WSQUOTA.

Необязательный выходной код элемента:

SJC$_ENTRY_NUMBER_OUTPUT;

SJC$_CREATE_QUEUE - данный запрос создает очередь. Если такая очередь уже существует и не остановлена, то данный запрос не выполняет никаких операций. Однако, если такая очередь уже существует и остановлена, то запрос изме-

няет параметры очереди на основании кодов элемента, заданных в запросе; если задан код элемента SJC$_CREATE_START, то запрос запускает данную очередь.

Обязательный входной код элемента:

SJC$_QUEUE.

Необязательные входные коды элемента:

1) SJC$_BASE_PRIORITY;

2) SJC$_BATCH;

3) SJC$_NO_BATCH;

4) SJC$_CHARACTERISTIC_NAME;

5) SJC$_NO_CHARACTERISTICS;

6) SJC$_CHARACTERISTIC_NUMBER;

7) SJC$_CPU_DEFAULT;

8) SJC$_NO_CPU_DEFAULT;

9) SJC$_CPU_LIMIT;

10) SJC$_NO_CPU_LIMIT;

11) SJC$_CREATE_START;

12) SJC$_DEVICE_NAME;

13) SJC$_FILE_BURST;

14) SJC$_NO_FILE_BURST;

15) SJC$_FILE_BURST_ONE;

16) SJC$_FILE_FLAG;

17) SJC$_NO_FILE_FLAG;

18) SJC$_FILE_FLAG_ONE;

19) SJC$_FILE_TRAILER;

20) SJC$_NO_FILE_TRAILER;

21) SJC$_FILE_TRAILER_ONE;

```
22) SJC$_FORM_NAME;
23) SJC$_FORM_NUMBER;
24) SJC$_GENERIC_QUEUE;
25) SJC$_NO_GENERIC_QUEUE;
26) SJC$_GENERIC_SELECTION;
27) SJC$_NO_GENERIC_SELECTION;;
28) SJC$_GENERIC_TARGET;
29) SJC$_JOB_BURST;
30) SJC$_NO_JOB_BURST;
31) SJC$_JOB_FLAG;
32) SJC$_NO_JOB_FLAG;
33) SJC$_JOB_LIMIT;
34) SJC$_JOB_RESET_MODULES;
35) SJC$_NO_JOB_RESET_MODULES;
36) SJC$_JOB_SIZE_MAXIMUM;
37) SJC$_NO_JOB_SIZE_MAXIMUM;
38) SJC$_JOB_SIZE_MINIMUM;
39) SJC$_NO_JOB_SIZE_MINIMUM;
40) SJC$_JOB_SIZE_SCHEDULIN;
41) SJC$_NO_JOB_SIZE_SCHEDULING;
42) SJC$_JOB_TRAILER;
43) SJC$_NO_JOB_TRAILER;
44) SJC$_LIBRARY_SPECIFICATION;
45) SJC$_NO_LIBRARY_SPECIFICATION;
46) SJC$_OWNER_UIC;    -
47) SJC$_PAGINATE;
48) SJC$_NO_PAGINATE;
```

49) SJC$_PROCESSOR;

50) SJC$_NO_PROCESSOR;

51) SJC$_PROTECTION;

52) SJC$_RECORD_BLOCKING;

53) SJC$_NO_RECORD_BLOCKING;

54) SJC$_RETAIN_ALL_JOBS;

55) SJC$_NO_RETAIN_JOBS;

56) SJC$_RETAIN_ERROR_JOBS;

57) SJC$_SCSNODE_NAME;

58) SJC$_SWAP;

59) SJC$_NO_SWAP;

60) SJC$_TERMINAL;

61) SJC$_NO_TERMINAL;;

62) SJC$_WSDEFAULT;

63) SJC$_NO_WSDEFAULT;

64) SJC$_WSEXTENT;

65) SJC$_NO_WSEXTENT;

66) SJC$_WSQUOTA;

67) SJC$_NO_WSQUOTA.

SJC$_DEASSIGN_QUEUE - данный запрос отменяет логическую очередь для очереди выполнения.

Обязательный входной код элемента:

SJC$_QUEUE;

SJC$_DEFINE_CHARACTERISTIC - данный запрос определяет имя и номер характеристики и вставляет данное определение в файл очереди. Каждое имя характеристики имеет уникальный номер в диапазоне от 0 до 127. Если такое имя характеристи-

ки уже определено, то запрос изменяет определение данной характеристики.

Задание не может выполняться в той же очереди выполнения, которая не обладает всеми характеристиками, принадлежащими данному заданию; но очередь может обладать дополнительными характеристиками, и задание будет выполняться.

Обязательные входные коды элементов:

1) SJC¤_CHARACTERISTIC_NAME;

2) SJC¤_CHARACTERISTIC_NUMBER;

SJC¤_DEFINE_FORM - данный запрос определяет имя и номер формы наряду с другими физическими атрибутами ассортимента бумаги и вставляет данное определение в системный файл очереди задания. Если такое имя формы уже определено, то запрос изменяет определение данной формы.

Формы используются только выходными очередями выполнения и печатающими заданиями. Печатающее задение не может выполниться, если имя ассортимента, заданное для очереди, отличается от имени ассортимента, заданного для данного задания. Имя ассортимента, принадлежащее форме, которое задано при помощи кода элемента SJC¤_FORM_STOCK, указывает ассортимент бумаги, используемой АЦПУ. Другие коды элемента задают такие параметры печати для задания, как поля, длину бумаги и т.п.

Каждая форма должна иметь уникальный номер. Когда создается новый файл очереди, система обеспечивает определение формы по имени DEFAULT с номером 0 и принимаемые по умолчанию характеристики.

Обязательные входные коды элемента:

1) SJC¤_FORM_NAME;

2) SJC¤_FORM_NUMBER.

Необязательные входные коды элемента:

1) SJC¤_FORM_DESCRIPTION;

2) SJC¤_FORM_LENGTH;

3) SJC¤_FORM_MARGIN_BOTTOM;

4) SJC¤_FORM_MARGIN_LEFT;

5) SJC¤_FORM_MARGIN_RIGHT;

6) SJC¤_FORM_MARGIN_TOP;

7) SJC¤_FORM_SETUP_MODULES;

8) SJC¤_NO_FORM_SETUP_MODULES;

9) SJC¤_FORM_SHEET_FEED;

10) SJC¤_NO_FORM_SHEET_FEED;

11) SJC¤_FORM_STOCK;

12) SJC¤_FORM_TRUNCATE;

13) SJC¤_NO_FORM_TRUNCATE;

14) SJC¤_FORM_WIDTH;

15) SJC¤_FORM_WRAP;

16) SJC¤_NO_FORM_WRAP;

17) SJC¤_PAGE_SETUP_MODULES;

18) SJC¤_NO_PAGE_SETUP_MODULES;

SJC¤_DELETE_CHARACTERISTIC - данный запрос удаляет определение имени характеристики.

Обязательный входной код элемента:

SJC¤_CHARACTERISTIC_NAME;

SJC¤_DELETE_FORM - данный запрос удаляет задание из системного файла очереди или задания, в которых есть обращение к данной форме.

Обязательный входной код элемента:

SJC¤_FORM_NAME;

SJC¤_DELETE_JOB - данный запрос удаляет задание из системного файла очереди заданий. Если задание в настоящий момент выполняется, то оно прерывается.

Обязательные входные коды элемента:

1) SJC¤_QUEUE;

2) SJC¤_ENTRY_NUMBER;

SJC¤_DELETE_QUEUE - данный запрос удаляет очередь и все задания в очереди. Данная очередь должна быть остановленной, и не должно существовать других очередей или заданий, в которых есть обращение к данной очереди.

Обязательный входной код элемента:

SJC¤_QUEUE;

SJC¤_ENTER_FILE - данный запрос создает задание, содержащее один файл, и помещает данное задание в указанную очередь. Для того, чтобы создать задание с несколькими файлами, нужно последовательно вызывать программу системного обслуживания ¤SNDJBC, используя коды функции SJC¤_CREATE_JOB, SJC¤_ADD_FILE и SJC¤_CLOSE_JOB.

Обязательный входной код элемента:

SJC$_QUEUE.

Обязателен один из следующих входных кодов элемента:

1) SJC$_FILE_IDENTIFICATION;

2) SJC$_FILE_SPECIFICATION.

Необязательные входные коды элемента:

1) SJC$_ACCOUNT_NAME;

2) SJC$_AFTER_TIME;

3) SJC$_NO_AFTER_TIME;

4) SJC$_CHARACTERISTIC_NAME;

5) SJC$_NO_CHARACTERISTICS;

6) SJC$_CHARACTERISTIC_NUMBER;

7) SJC$_CLI;

8) SJC$_NO_CLI;

9) SJC$_CPU_LIMIT;

10) SJC$_NO_CPU_LIMIT;

11) SJC$_DELETE_FILE;

12) SJC$_NO_DELETE_FILE;

13) SJC$_DOUBLE_SPACE;

14) SJC$_NO_DOUBLE_SPACE;

15) SJC$_FILE_BURST;

16) SJC$_NO_FILE_BURST;

17) SJC$_FILE_COPIES;

18) SJC$_FILE_FLAG;

19) SJC$_NO_FILE_FLAG;

20) SJC$_FILE_SETUP_MODULES;

21) SJC$_NO_FILE_SETUP_MODULES;

```
22) SJC¤_FILE_TRAILER;
23) SJC¤_NO_FILE_TRAILER;
24) SJC¤_FIRST_PAGE;
25) SJC¤_NO_FIRST_PAGE;
26) SJC¤_FORM_NAME;
27) SJC¤_FORM_NUMBER;
28) SJC¤_HOLD;
29) SJC¤_NO_HOLD;
30) SJC¤_JOB_COPIES;
31) SJC¤_JOB_NAME;
32) SJC¤_LAST_PAGE;
33) SJC¤_NO_LAST_PAGE;
34) SJC¤_LOG_DELETE;
35) SJC¤_NO_LOG_DELETE;
36) SJC¤_LOG_QUEUE;
37) SJC¤_LOG_SPECIFICATION;
38) SJC¤_NO_LOG_SPECIFICATION;
39) SJC¤_SPOOL;
40) SJC¤_NO_SPOOL;
41) SJC¤_LOWERCASE;
42) SJC¤_NO_LOWERCASE;
43) SJC¤_NOTE;
44) SJC¤_NO_NOTE;
45) SJC¤_NOTIFY;
46) SJC¤_NO_NOTIFY;
47) SJC¤_OPERATOR_REQUEST;
48) SJC¤_NO_OPERATOR_REQUEST;
```

49) SJC$_PAGE_HEADER;

50) SJC$_NO_PAGE_HEADER;

51) SJC$_PAGINATE;

52) SJC$_NO_PAGINATE;

52) SJC$_PARAMETER_1 до 8;

54) SJC$_NO_PARAMETERS;

55) SJC$_PASSALL;

56) SJC$_NO_PASSALL;

57) SJC$_PRIORITY;

58) SJC$_RESTART;

59) SJC$_NO_RESTART;

60) SJC$_UIC;

61) SJC$_USERNAME;

62) SJC$_WSDEFAULT;

63) SJC$_NO_WSDEFAULT;

64) SJC$_WSEXTENT;

65) SJC$_NO_WSEXTENT;

66) SJC$_WSQUOTA;

67) SJC$_NO_WSQUOTA.

Необязательные выходные коды элемента:

1) SJC$_ENTRY_NUMBER_OUTPUT;

2) SJC$_JOB_STATUS_OUTPUT;

SJC$_MERGE_QUEUE - данный запрос предоставляет все задания из очереди, указанной кодом элемента SJC$_QUEUE, в свою очередь указанную кодом элемента SJC$_DESTINATION_QUEUE. На выполнение текущих заданий это не влияет.

Обязательные входные коды элементов:

1) SJC¤_QUEUE;

2) SJC¤_DESTINATION_QUEUE;

SJC¤_PAUSE_QUEUE - данный запрос приостанавливает выполнение текущих заданий в указанной очереди и прекращает запуск заданий в данной очереди.

Обязательный входной код элемента:

SJC¤_QUEUE;

SJC¤_RESET_QUEUE - данный запрос сбрасывает указанную очередь следующим образом: заканчивает и удаляет каждое выполняющееся задание, которое не является повторно запускаемым, заканчивает и снова ставит в очередь каждое выполняющееся задание, которое является повторно запускаемым и останавливает очередь.

Обязательный входной код элемента:

SJC¤_QUEUE;

SJC¤_START_ACCOUNTING - данный запрос выполняет две функции. Если задан код элемента SJC¤_ACCOUNTING_TYPES, то запрос разрешает регистрацию учетных записей указанных типов; если SJC¤_ACCOUNTING_TYPES не задан, то данный запрос запускает планировщик учетной информации и открывает системный учетный файл.

Обязателен один из следующих кодов элемента:

1) SJC¤_ACCOUNTING_TYPES;

2) SJC¤_NEW_VERSION;

SJC¤_START_QUEUE - данный запрос разрешает запуск заданий в указанной очереди. Если очередь была приостанов-

лена, то возобновляются текущие задачия.

Обязательный входной код элемента:

SJC$_QUEUE.

Необязательные входные коды элемента:

1) SJC$_ALIGNMENT_MASK;

2) SJC$_ALIGNMENT_PAGES;

3) SJC$_BASE_PRIORITY;

4) SJC$_BATCH;

5) SJC$_NO_BATCH;

6) SJC$_CHARACTERISTIC_NAME;

7) SJC$_NO_CHARACTERISTICS;

8) SJC$_CHARACTERISTIC_NUMBER;

9) SJC$_CPU_DEFAULT;

10) SJC$_NO_CPU_DEFAULT;

11) SJC$_CPU_LIMIT;

12) SJC$_NO_CPU_LIMIT;

13) SJC$_DEVICE_NAME;

14) SJC$_FILE_BURST;

15) SJC$_NO_FILE_BURST;

16) SJC$_FILE_BURST_ONE;

17) SJC$_FILE_FLAG;

18) SJC$_NO_FILE_FLAG;

19) SJC$_FILE_FLAG_ONE;

20) SJC$_FILE_TRAILER;

21) SJC$_NO_FILE_TRAILER;

22) SJC$_FILE_TRAILER_ONE;

23) SJC$_FORM_NAME;

```
24) SJC$_FORM_NUMBER;

25) SJC$_GENERIC_QUEUE;

26) SJC$_NO_GENERIC_QUEUE;

27) SJC$_GENERIC_SELECTION;

28) SJC$_NO_GENERIC_SELECTION;

29) SJC$_GENERIC_TARGET;

30) SJC$_JOB_BURST;

31) SJC$_NO_JOB_BURST;

32) SJC$_JOB_FLAG;

33) SJC$_NO_JOB_FLAG;

34) SJC$_JOB_LIMIT;

35) SJC$_JOB_RESET_MODULES;

36) SJC$_NO_JOB_RESET_MODULES;

37) SJC$_JOB_SIZE_MAXIMUM;

38) SJC$_NO_JOB_SIZE_MAXIMUM;

39) SJC$_JOB_SIZE_MINIMUM;

40) SJC$_NO_JOB_SIZE_MINIMUM;

41) SJC$_JOB_SIZE_SCHEDULING;

42) SJC$_NO_JOB_SIZE_SCHEDULING;

43) SJC$_JOB_TRAILER;

44) SJC$_NO_JOB_TRAILER;

45) SJC$_LIBRARY_SPECIFICATION;

46) SJC$_NO_LIBRARY_SPECIFICATION;

47) SJC$_NEXT_JOB;

48) SJC$_OWNER_UIC;

49) SJC$_PAGINATE;

50) SJC$_NO_PAGINATE;
```

51) SJC$_PROCESSOR;

52) SJC$_NO_PROCESSOR;

53) SJC$_PROTECTION;

54) SJC$_RECORD_BLOCKING;

55) SJC$_NO_RECORD_BLOCKING;

56) SJC$_RELATIVE_PAGE;

57) SJC$_RETAIN_ALL_JOBS;

58) SJC$_NO_RETAIN_JOBS;

59) SJC$_RETAIN_ERROR_JOBS;

60) SJC$_SEARCH_STRING;

61) SJC$_SWAP;

62) SJC$_NO_SWAP;

63) SJC$_TERMINAL;

64) SJC$_NO_TERMINAL;

65) SJC$_TOP_OF_FILE;

66) SJC$_WSDEFAULT;

67) SJC$_NO_WSDEFAULT;

68) SJC$_WSEXTENT;

69) SJC$_NO_WSEXTENT;

70) SJC$_WSQUOTA;

71) SJC$_NO_WSQUOTA;

SJC$_START_QUEUE_MANAGER - данный запрос запускает планировщик очередей, применяя принятые по умолчанию спецификации файла из SYS$SYSTEM:JBCSYSQUE.DAT; либо он открывает существующий системный файл очереди заданий, либо создает новый. Использование кода элемента SJC$_$NEW_VERSION приводит к созданию нового системного файла очереди зада-

ний.

Необязательные входные коды элементов:

1) SJC$_BUFFER_COUNT;

2) SJC$_EXTENT_QUANTITY;

3) SJC$_QUEUE_FILE_SPECIFICATION;

4) SJC$_NEW_VERSION;

SJC$_STOP_ACCOUNTING - данный запрос выполняет две функции. Если задан код элемента SJC$_ACCOUNTING_TYPES, то данный запрос запрещает регистрацию учетных записей указанных типов. Если SJC$_ACCOUNTING_TYPES не задан, то запрос останавливает планировщик учетной информации и закрывает системный учетный файл.

Необязательный входной код элемента:

SJC$_ACCOUNTING_TYPES;

SJC$_STOP_QUEUE - данный запрос запрещает запуск заданий в указанной очереди. На выполнение текущих заданий это не влияет.

Обязательный входной код элемента:

SJC$_QUEUE;

SJC$_STOP_QUEUE_MANAGER - данный запрос завершает выполнение планировщика очередей. Он останавливает свою очередь; прерывает каждое выполняющееся в текущий момент задание, снова помещая в очередь те задания, которые являются повторно запускаемыми; и закрывает системный файл очереди заданий. Никакие коды элементов не допускаются;

SJC$_SYNCHRONIZE_JOB - данный запрос ожидает завершения какого-либо задания, затем устанавливает флаг события,

об'являет AST завершения и возвращает код завершения данного задания в блок состояния ввода-вывода при условии, что в вызове программы системного обслуживания $SNDJBC был задан аргумент IOSB.

Обязательный входной код элемента:

SJC$_QUEUE.

Обязателен один из следующих входных кодов элемента:

1) SJC$_ENTRY_NUMBER;

2) SJC$_JOB_NAME;

SJC$_WRITE_ACCOUNTING - данный запрос записывает учетную запись.

Обязательный входной код элемента:

SJC$_ACCOUNTING_MESSAGE.

NULLARG

Использование в МОС ВП: NULL_ARG

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Данный аргумент не используется.

ITMLST

Использование в МОС ВП: ITEM_LIST_3

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Список элементов, обеспечивающих информацию, которая должна использоваться при выполнении функции, заданной аргументом FUNC. Аргумент ITMLST - это адрес списка элемен-

тов. Список элементов состоит из одного или нескольких дескрипторов элементов, каждый из которых определяет код элемента. Список элементов заканчивается кодом элемента 0 или длинным словом, содержащим 0. Рис.80 показывает структуру одного дескриптора элемента.

Дескриптор элемента

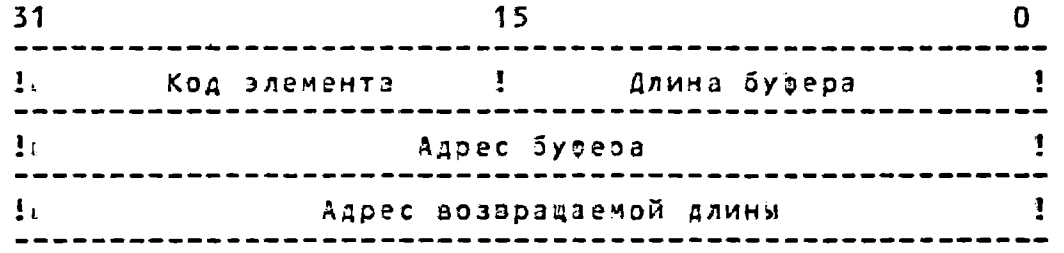


Рис. 80

Поля дескриптора элемента ¤SNDJBC:

длина буфера -

Слово, задающее длину буфера; данный буфер либо обеспечивает информацию, которую использует ¤SNDJBC, либо получает информацию от ¤SNDJBC. Требуемая длина буфера изменяется в зависимости от указанного кода элемента и задается в описании каждого кода элемента;

код элемента -

Слово, содержащее код элемента, который определяет тип информации, обеспечиваемой для ¤SNDJBC или получаемой от ¤SNDJBC. Каждый код элемента имеет символическое имя; данные имена определяются при помощи макрокоманды ¤SJCDEF и имеют формат: SJC¤_код.

Существуют три типа кодов элемента:

1) булев код элемента. Булевы коды элемента задают значение "истина" или "ложь"; формат SJC_код задает значение "истина", а формат SJC¤_NO_код задает значение "ложь". Для булевых кодов элемента поля дескриптора элемента "длина буфера", "адрес буфера" и "возвращаемая длина" должны быть нулевыми;

2) входной код элемента. Входные коды элемента задают входные значения, которые используются программой ¤SNDJBC. Для входных кодов элемента, поля дескриптора элемента "длина буфера" и "адрес буфера" должны быть ненулевыми; поле "адрес возвращаемой длины" должно быть нулевым. Конкретные требования к длине буфера задаются в описании каждого кода элемента;

3) выходной код элемента. Выходные коды элемента определяют буфер для информации, возвращаемой программой ¤SNDJBC. Для выходных кодов элемента поля дескриптора элемента "длина буфера" и "адрес буфера" должны быть ненулевыми; поле "адрес возвращаемой длины" может быть нулевым или ненулевым. Конкретные требования к длине буфера задаются в описании каждого кода элемента.

Несколько кодов элемента задают имя очереди, имя формы или имя характеристики. Для таких кодов элемента буфер должен задавать строку, содержащую от 1 до 31 символа, исключая пробелы, знаки табуляции и пустые символы, которые игнорируются. Допустимыми символами в строке являются прописные буквенные символы, строчные буквенные символы (которые преобразуются в прописные), цифровые символы, знак

денежной единицы (¤) и знак подчеркивания (_);

адрес буфера -

Адрес буфера, который задает или получает информацию;

адрес возвращаемой длины -

Адрес слова, которое получает длину информации (в байтах), возвращаемой программой ¤SNDJBC. Если данный адрес был задан равным 0, то длина не возвращается.

Коды элементов программы ¤SNDJBC:

SJC¤_ACCOUNT_NAME - это входной код элемента. Данное значение задает 8-байтовое учетное имя пользователя, для которого делается данный запрос. По умолчанию учетное имя берется от запрашивающего процесса. Данный код требует привилегии CMKRNL;

SJC¤_ACCOUNTING_MESSAGE -

Это входной код элемента. Он заставляет поместить содержимое буфера в учетную запись "пользовательские данные". Буфер должен задавать строку длиной от 1 до 255 символов;

SJC¤_ACCOUNTING_TYPES -

Это входной код элемента. Он разрешает или запрещает определенные типы учетных записей. Если какой-то тип учетной записи разрешен, то событие, обозначенное данным типом, будет занесено в данную учетную запись. Буфер должен содержать битовую маску размером в длинное слово, где каждый бит определяет тип учетной записи. Неиспользованные биты должны быть нулевыми.

такие типы имеют символические имена, которые опреде-

ляют при помощи макрокоманды ¤SJCDEF. Битовая маска размером в длинное словов строится при помощи операции "логическое или", выполненной для символических имен всех требуемых типов учетных записей.

Типы учетных записей и системные события, которые им соответствуют:

1) SJC¤V_ACCT_IMAGE - завершение образа;

2) SJC¤V_ACCT_LOGIN_FAILURE - ошибки при подключении;

3) SJC¤V_ACCT_MESSAGE - посланные пользователем сообщения;

4) SJC¤V_ACCT_PRINT - завершение печатающих заданий;

5) SJC¤V_ACCT_PROCESS - завершение процесса.

Следующие типы учетных записей, когда они разрешены, обеспечивают дополнительную информацию о завершении образа и процесса; а именно, они описывают тип образа или процесса, который завершился:

1) SJC¤V_ACCT_BATCH - пакетный процесс;

2) SJC¤V_ACCT_DETACHED - отсоединенный процесс;

3) SJC¤V_ACCT_INTERACTIVE - интерактивный процесс;

4) SJC¤V_ACCT_NETWORK - сетевой процесс;

5) SJC¤V_ACCT_SUBPROCESS - подпроцесс;

SJC¤_AFTER_TIME -

Это входной код элемента. Он указывает, что задание может выполняться только в том случае, если системное время больше или равно значению времени в квадрослове, содежащемуся в буфере. Буфер должен содержать либо значение абсолютного времечи, либо значение

дельта-времени; $SNDJBC преобразует значение дельта-времени в значение абсолютного времени, прибавляя текущее системное время;

SJC$_NO_AFTER_TIME -

Это булев код элемента. Он указывает, что задание может выполняться немедленно. Данный код принят по умолчанию;

SJC$_ALIGNMENT_MASK -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения и только в том случае, если задан код элемента SJC$_ALIGNMENT_PAGES. SJC$_ALIGNMENT_MASK означает, что данные, которые печатаются на страницах выравнивания форм, должны быть замаскированы: все буквенные символы заменяются буквой "X", а все цифровые символы - цифрой 9;

SJC$_ALIGNMENT_PAGES -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Данный код указывает, что очередь необходимо перевести в состояние выравнивания форм, и надо напечатать некоторое количество страниц выравнивания. Буфер должен содержать значение в диапазоне от 1 до 20 размером в длинное слово; данное значение указывает, сколько страниц выравнивания требуется напечатать;

SJC$_BASE_PRIORITY -

Это входной код элемента. Он имеет значение только для очередей выполнения. Данный код задает базовый приори-

тет для пакетных заданий, инициированных из пакетной очереди, или для симбионтных процессов, связанных с выходной очередью выполнения. Симбионтный процесс может управлять несколькими очередями; однако, базовый приоритет симбионтного процесса устанавливается первой очередью, с которой он связан. Буфер должен содержать значение размером в длинное слово в диапазоне от 1 до 15; данное значение задает базовый приоритет. По умолчанию базовым приоритетом является значение параметра системной генерации DEFPRI. Если значение DEFPRI равно 0, то по умолчанию базовый приоритет равен базовому приоритету запрашивающего процесса;

SJC$_BATCH -

Это булев код элемента. Он указывает, что очередь является пакетной очередью для выполнения или общей пакетной очередью, и, таким образом, может обрабатывать только печатающие задания. Данный код принят по умолчанию;

SJC$_BUFFER_COUNT -

Это входной код элемента. Он задает количество буферов, которое диспетчер заданий должен выделить своему кэшу локальных буферов для выполнения операций ввода-вывода в системный файл очереди заданий. Буфер данного кода должен содержать целое значение размером в длинное слово в диапазоне от 1 до 127 или 0; данное значение определяет количество буферов, которое диспетчер заданий выделяет для своего кэша локальных

буферов. Если задан 0, то используется принимаемое по умолчанию значение 50;

SJC$_CHARACTERISTIC_NAME и SJC$_CHARACTERISTIC_NUMBER - Это входные коды элементов. Оба определяют характеристики заданий или очереди и могут использоваться взаимозаменяемо. Характеристики определяет пользователь. Коды функций SJC$_DEFINE_CHARACTERISTIC и SJC$_DELETE_CHARACTERISTIC соответственно включают и удаляют заданные характеристики из системного файла очереди заданий. Задание не может работать в очереди выполняющихся заданий, если данная очередь не обладает всеми характиристиками, присущими заданию; но очередь может иметь дополнительные характеристики, и задание будет выполняться. Коды элементов SJC$_CHARACTERISTIC_NAME и SJC$_CHARACTERISTIC_NUMBER могут появлятся в одном вызове столько раз, сколько это необходимо; полученый таким образом в вызове набор характеристик замещает набор характеристик, определенный предыдущим вызовом. По умолчанию очередь или задание не имеет определенных характеристик. Для кода SJC$_CHARACTERISTIC_NAME строка имени может содержать прописные или строчные символы (строчные преобразуются в прописные), цифровые символы, знаки денежной единицы и подчеркивания. Если строка является логическим именем, то SYS$SNDJBC итеративно транслирует его до тех пор, пока не будет найдена эквивалентная строка или пока количество трансляций не достигнет значения,

допускаемого системой. Максимальная длина окончательной символьной строки - 31 символ; пробелы, знаки табуляции и пустые символы игнорируются. Для кода SJC$_CHARACTERISTIC_NUMBER буфер должен содержать значение в диапазоне от 0 до 127 размером в длинное слово. Данный номер идентифицирует характеристику;

SJC$_NO_CHARACTERISTIC -

Это булев код элемента. Он отменяет все определенные для данного задания или очереди характеристики;

SJC$_CHECKPOINT_DATA -

Это входной код элемента. Он задает для пакетного задания, которое является повторно запускаемым, значение символа DCL BATCH$RESTART. Буфер должен содержать строку длиной не более 255 символов; данная строка является значением символа BATCH_RESTART;

SJC$_NO_CHECKPOINT_DATA -

Это булев код элемента. Он отменяет предыдущее определение символа BATCH$RESTART; он также отменяет контрольную точку в печатающем задании, так что снова печатается все задание целиком. По умолчанию символ BATCH$RESTART не определен;

SJC$_CLI -

Это входной код элемента. Он имеет значение только для паке данный код указывает, что интерпретатором языка команд, подлежащим выполнению, является SYS$SYSTEM:имя.EXE, где "имя" - это допустимое имя файла. Буфер должен задавать строку имечи длиной от 1

до 39 символов;

SJC$_NO_CLI -

Это булев код элемента. Он указывает, что необходимо выполнять интерпретатор языка команд, заданный в файле авторизации пользователя. Данный код принимается по умолчанию;

SJC$_CPU_DEFAULT -

Это входной код элемента. Он имеет значение только для пакетных очередей выполнения. Данный код задает принимаемый по умолчанию предел времени процессора в 10-миллисекундных единицах. Буфер содержит данное значение размером в длинное слово. Значение 0 указывает на неограниченное время процессора;

SJC$_NO_CPU_DEFAULT -

Это булев код элемента. Он имеет значение только для пакетных очередей выполнения. Данный код указывает, что к данному заданию не применяется принятый по умолчанию предел времени процессора. Данный код принимается по умолчанию.

Предел времени процессора для процесса включается в каждую пользовательскую запись в системном файле авторизации пользователя (UAF). Кроме того, можно задать некоторые или все из следующих значений: предел времени процессора для отдельных заданий, принимаемый по умолчанию, предел времени процессора для всех заданий в данной очереди и предел максимального времени процессора для всех заданий в данной очереди. В табл.69 описаны действия, которые предп-

ринимаются, если для SJC$_CPU_DEFAULT задано какое-то значение.

Таблица 69

Предел времени процесса для задания указан?	Принимаемый по умолчанию предел времени процесса очереди указан?	Максимальное время процесса для очереди указано?	Предпринимаемые действия
нет	нет	нет	используется значение UAF
да	нет	нет	используется меньшее из значений предела задания и UAF
да	да	нет	используется меньшее из значений задания и UAF
да	нет	да	используется меньшее из предела задания и максимального значения
да	да	да	используется меньшее из предела задания и максимального значения
нет	да	да	используется меньшее из принятого по умолчанию и максимального значений для очереди

Продолжение табл. 69

Предел. времени процес-сора для за-дания указан?	Принимаемый по умолчанию предел вре-мени процес-сора очереди указан?	Максима-льное время процесс-сора для очереди указано?	Предпринимаемые действия
нет	нет	да	используется максимальное значение
нет	да	нет	используется меньшее из значения UAF и принятого по умолчанию значения для очереди

SJC$_CPU_LIMIT -

Это входной код элемента. Он имеет значение только для пакетных очередей выполнения и для пакетных заданий. Данный код задает максимальный предел времени процессора в 10-миллисекундных единицах. Буфер должен содержать данное значение размером в длинное слово. Значение 0 указывает на неограниченное время процессора;

SJC$_NO_CPU_LIMIT -

Это булев код элемента. Он имеет значение только для пакетных очередей выполнения и для пакетных заданий. он указывает, что к данному заданию неприменим макси-

мальный предел времени процессора. Данный код принимается по умолчанию.

Информация о действиях, которые предпринимаются, если для SJC¤_CPU_LIMIT задано какое-то значение, приводится в описании кода элемента SJC¤_CPU_DEFAULT и в табл.69;

SJC¤_CREATE_START -

Это булев код элемента. Он указывает на то, что очередь можно запустить после того, как она будет создана. По умолчанию очередь после ее создания остается остановленной;

SJC¤_DELETE_FILE -

Это булев код элемента. Он указывает, что после завершения задания файл необходимо удалить. Данный код элемента нельзя указывать с кодом функции SJC¤_ALTER_JOB, который изменяет параметры уже существующего задания; запрос на удаление файла можно сделать только в том случае, когда задание впервые ставится в очередь;

SJC¤_NO_DELETE_FILE -

Это булев код элемента. Он указывает, что после выполнения задания файл не необходимо удалять. Это принимается по умолчанию. Данный код элемента можно задавать с кодом функции SJC¤_ALTER_JOB; это дает возможность отменить запрос на удаление файла, который был сделан, когда данное задание впервые ставилось в очередь;

SJC¤_DESTINATION_QUEUE -

Это входной код элемента. Когда указан код функции

SJC¤_ASSIGN_QUEUE, SJC¤_DESTINATION_QUEUE задает имя очереди выполнения, которой назначается логическая очередь. Когда указаны коды функции SJC¤_ABORT_JOB, SJC¤_ALTER_JOB или SJC¤_MERGE_QUEUE, код элемента SJC¤_DESTINATION_QUEUE задает имя той очереди, в которую помещаются задания. По умолчанию задания остаются в исходной очереди.

Строка имени может содержать прописные или строчные символы (строчные преобразуются в прописные), цифровые символы, знаки денежной единицы и подчеркивания. Если строка является логическим именем, SYS¤SNDJBC итеративно транслирует до тех пор, пока не будет найдена эквивалентная строка или пока не выполнится такое количество трансляций, которое допускается операционной системой MOC ВП. Максимальная длина окончательной символьной строки - 31 символ; пробелы, знаки табуляции и пустые символы игнорируются;

SJC¤_DEVICE_NAME -

Это входной код элемента. Он имеет значение только для очередей заданий. Данный код задает узел и/или устройство, на котором размещается указанная очередь выполнения. Для пакетных очередей можно указывать только имя узла. Буфер должен задавать строку длиной от 1 до 31 символа;

SJC¤_DOUBLE_SPACE -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Данный код указывает, что выходной симбионт должен печатать один файл через

два интервала. По умолчанию файл печатается через один интервал;

SJC$_ENTRY_NUMBER -

Это выходной код элемента. Он задает номер входа задания, для которого выполняется данная функция. Буфер должен содержать данный номер входа;

SJC$_ENTRY_NUMBER_OUTPUT -

Это выходной код элемента. Буфер должен содержать длинное слово, в которое $SNDJBC запишет номер входа созданного задания;

SJC$_EXTENT_QUANTITY -

Это входной код элемента. Он задает размер экстента системного файла очереди заданий в блоках. Данный размер экстента используется при создании файла очередей для определния начального размера распределения. Буфер должен содержать целое значение размером в длинное слово в диапазоне от 10 до 65535 или 0. Данное значение определяет количество блоков, на которое следует расширять очередь. По умолчанию принимается значение 100 блоков. Если задано значение 0, то используется принятый по умолчанию размер;

SJC$_FILE_BURST -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Данный код указывает, что перед файлом требуется напечатать страницу разрыва. Если код SJC$_FILE_BURST указан для задания, то по умолчанию он распространяется на все файлы данного

задания; если он указан для входной очереди выполнения, то по умолчанию его действие распространяется на все задания, выполняемые из данной очереди;

SJC$_FILE_BURST_ONE -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Данный код указывает, что перед файлом необходимо напечатать страницу разрыва. Если код SJC$_FILE_BURST_ONE указан для задания, то он означает, что страницу разрыва надо печатать только перед первой копией первого файла в задании; если код задан для выходной очереди выполнения, то по умолчанию он определяет действия для всех заданий, выполняемых из данной очереди;

SJC$_NO_FILE_BURST -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он указывает, что страницу разрыва печатать не требуется. Данный код принят по умолчанию;

SJC$_FILE_COPYES -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Он указывает, сколько раз необходимо повторить файл. По умолчанию файл повторяется один раз. Буфер должен содержать значение от 1 до 255 размером в длинное слово; данное значение является счетчиком повторений;

SJC$_FILE_FLAG -

Это булев код элемента. Он имеет значение только для

выходных очередей выполнения. Он указывает, что перед файлом необходимо печатать страницы флагов. Если код SJC$_FILE_FLAG указан для задания, то по умолчанию распространяется на все файлы в данном задании; если код задан для выходной очереди выполнения, то по умолчанию он распространяется на все задания, выполненные из данной очереди;

SJC$_FILE_FLAG_ONE -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он указывает, что перед файлом необходимо напечатать страницу флагов. Если код SJC$_FILE_FLAG_ONE указан для задания, то он обозначает, что страницу флага задания необходимо печатать только перед первой копией первого файла в задании; если код указан для выходной очереди выполнения, то по умолчанию он определяет такие действия для всех заданий, выполненных из данной очереди;

SJC$_NO_FILE_FLAG -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он указывает, что страницу флагов печатать не требуется. Данный код принят по умолчанию;

SJC$_FILE_IDENTIFICATION -

Это входной код элемента. Он задает файл, подлежащий обработке. Буфер содержит 28-байтовое значение, которое идентифицирует файл, подлежащий обработке. Данное значение задает (по порядку) следующие три поля:

16-байтовое поле NAM¤T_DVI, 6-байтовое поле NAM¤W_FID и 6-байтовое поле NAM¤W_DID. Данные поля располагаются в блоке NAM последовательно и в том порядке, как они перечислены.

Если задан код SJC¤_FILE_IDENTIFICATION, то код элемента SJC¤_FILE_SPECIFICATION должен быть опущен;

SJC¤_FILE_SETUP_MODULES -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Он указывает, что перед тем, как печатать файл, необходимо извлечь из управляющей библиотеки устройства список текстовых модулей и скопировать их на АЦПУ. Буфер должен содержать список имен текстовых модулей, разделенных запятыми;

SJC¤_NO_FILE_SETUP_MODULES -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он указывает, что перед печатью файла не требуется копировать текстовые модули. Данный код принят по умолчанию;

SJC¤_FILE_SPECIFICATION -

Это входной код элемента. Он идентифицирует файл, подлежащий обработке. Буфер должен содержать спецификацию файла, подлежащего обработке. Программа системного обслуживания ¤SNDJBC преобразует спецификацию файла в соответствующую идентификацию файла и дальше работает так, как если был указан код элемента SJC¤_FILE_IDENTIFICATION. Если задан код SJC¤_FILE_SPECIFICATION, то код элемента

SJC$_FILE_IDENTIFICATION должен быть опущен;

SJC$_FILE_TRAILER -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он указывает, что после файла необходимо печатать дополнительную страницу. Если код SJC$_FILE_TRAILER указан для задания, то по умолчанию он распространяется на все файлы в данном задании; если код указан для выходной очереди выполнения, то по умолчанию он распространяется на все задания, выполненные в данной очереди;

SJC$_FILE_TRAILER_ONE -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он указывает, что после файла требуется печатать дополнительную страниц; если код SJC$_FILE_TRAILER_ONE указан для задания, то он означает, что дополнительную страницу требуется печатать только за последней копией последнего файла в данном задании; если код указан для выходной очереди выполнения, то он по умолчанию определяет такие действия для всех заданий, выполненных из данной очереди;

SJC$_NO_FILE_TRAILER -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он означает, что дополнительную страницу печатать не требуется. Данный код принят по умолчанию;

SJC$_FIRST_PAGE -

Это выходной код элемента. Он имеет значение только

для выходных очередей выполнения. Он задает номер страницы, с которой требуется начать печатать. Буфер должен содержать ненулевое значение размером в длинное слово, задающее данный номер страницы;

SJC$_NO_FIRST_PAGE -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он означает, что печать требуется начинать с первой страницы. Данный код принят по умолчанию;

SJC$_FORM_DESCRIPTION -

Это входной код элемента. Он задает текст, описывающий данную форму для пользователей и операторов. По умолчанию используется имя формы. Буфер должен задавать строку длиной не более 255 символов;

SJC$_FORM_LENGTH -

Это входной код элемнта. Он имеет значение только для выходных очередей выполнения. Он задает физическую длину формы в строках. Буфер должен содержать ненулевое целое значение размером в длинное слово. По умолчанию длина формы - 66 строк;

SJC$_FORM_MARGIN_BOTTOM -

Это входной код элемента. Он задает нижнее поле формы в строках. По умолчанию нижнее поле составляет 6 строк;

SJC$_FORM_MARGIN_LEFT -

Это входной код элемента. Он задает левое поле формы в символах. По умолчанию поле равно 0. Буфер должен ука-

зывать значение размером в длинное слово;

SJC$_FORM_MARGIN_RIGHT -

Это входной код элемента. Он задает правое поле формы в символах. По умолчанию правое поле равно 0. Буфер должен указывать значение размером в длинное слово;

SJC$_FORM_MARGIN_TOP -

Это входной код элемента. Он задает верхнее поле формы в строках. По умолчанию поле разно 0;

SJC$_FORM_NAME и SJC$_FORM_NUMBER -

Это входные коды элементов. Они задают форму соответственно по имени и по номеру. Для SJC$_FORM_NAME буфер должен содержать имя формы. Для SJC$_FORM_NUMBER буфер должен содержать значение размером в длинное слово. Коды функций SJC$_DEFINE_FORM и SJC$__DELETE_FORM соответственно включают и удаляют указанное имя и номер формы из системного файла очереди заданий. Задание не может выполняться в выходной очереди, если заданная (по имени или по номеру) для данного задания форма не совпадает с именем ассортимента формы, заданным (по имени или по номеру) для данной очереди. По умолчанию форма имеет номер 0. Строка может содержать прописные или строчные символы (строчные преобразуются в прописные), цифровые символы, знаки денежной единицы и подчеркивания. Если строка является логическим именем, то $SNDJBC итеративно транслирует ее до тех пор, пока не будет найдено эквивалентное имя или пока количество выполненных трансляций не достигнет предела,

допустимого системой. Максимальная длина окончательной символьной строки - 31 символ; пробелы, знаки табуляции и пустые символы игнорируются;

SJC$_FORM_SETUP_MODULES -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Буфер должен задавать одно или несколько имен текстовых модулей, разделенных запятыми. Данный код элемента означает, что данные модули необходимо извлечь из управляющей библиотеки устройства и скопировать на ацпу перед каждым файлом, который печатается в данной форме;

SJC$_NO_FORM_SETUP_MODULES -

Это булев код элемента. Он означает, что управляющие модули устройства печатать не требуется. Данный код принят по умолчанию;

SJC$_FORM_SHEET_FEED -

Это булев код элемента. Он означает, что симбионт должен останавливаться в конце каждой физической страницы так, чтобы можно было вставить новую форму;

SJC$_NO_FORM_SHEET_FEED -

Это булев код элемента. Он означает, что выходной симбионт не должен останавливаться после каждой физической страницы. Данный код принят по умолчанию;

SJC$_FORM_STOCK -

Это выходной код элемента. Он задает имя для ассортимента бумаги. Буфер должен содержать строку длиной от 1 до 31 символа. По умолчанию именем ассортимента

бумаги является имя формы;

SJC$_FORM_TRUNCATE -

Это булев код элемента. Он означает, что симбионт должен отсекать строки, которые продолжаются за пределы правого поля. Задание кода SJC$_FORM_TRUNCATE отменяет действие кода SJC$_FORM_WRAP. Данный код принят по умолчанию;

SJC$_NO_FORM_TRUNCATE -

Это булев код элемента. Он означает, что выходной симбионт не должен отсекать строки, которые продолжаются за пределы правого поля;

SJC$_FORM_WIDTH -

Это входной код элемента. Он задает физическую ширину формы в символах. Буфер должен содержать ненулевое целое значение размером в длинное слово. По умолчанию ширина формы - 132 символа;

SJC$_FORM_WRAP -

Это булев код элемена. Он означает, что симбионт должен переносить строки, которые продолжаются за пределы правого поля. Задание кода SJC$_FORM_WRAP отменяет действие кода SJC$_FORM_TRUNCATE;

SJC$_FORM_NO_FORM_WRAP -

Это булев код элемента. Он означает, что выходной симбионт не должен переносить строки. Данный код принят по умолчанию;

SJC$_GENERIC_QUEUE -

Это булев код элемента. Он означает, что очередь

является общей очередью;

SJC$_NO_GENERIC_QUEUE -

Это булев код элемента. Он означает, что очередь не является общей очередью. Данный код принят по умолчанию. По умолчанию очередь является очередью выполнения;

SJC$_GENERIC_SELECTION -

Это булев код элемента. Он означает, что очередь выполнения может принимать задания из общей очереди. данный код принят по умолчанию. Он имеет значение только для очередей выполнения;

SJC$_NO_GENERIC_SELECTION -

Это булев код элемента. Он означает, что очередь выполнения не может принимать задания из общей очереди;

SJC$_GENERIC_TARGET -

Это входной код элемента. Буфер должен задавать имя очереди. Данное имя очереди идентифицирует очередь выполнения, которая может принимать задания из общей очереди. Данный код элемента имеет значение только для общих очередей. В одном вызове $SNDJBC можно задавать данный код элемента до 124 раз. Набор очередей, определенных при одном вызове, полностью замещает набор, определенный предыдущим вызовом. Строка имени может содержать прописные и строчные символы (строчные преобразуются в прописные), цифровые символы, знаки денежной единицы и подчеркивания. Если строка является

логическим именем, то SYS¤SNDJBC итеративно транслирует ее до тех пор, пока не будет найдена эквивалентная строка или количество выполненных трансляций не достигнет предела, допустимого в системе. Максимальная длина окончательной символьной строки - 31 символ; пробелы, знаки табуляции и пустые символы игнорируются;

SJC¤_HOLD -

Это булев код элемента. Он означает, что задание нельзя выполнить и его необходимо задержать;

SJC¤_NO_HOLD -

Это булев код элемента. Он означает, что задание можно выполнить немедленно; он также дает право на выполнение следующим типам заданий:

1) заданию, которое задержано на заданное время;

2) заданию, которое задержано по запросу симбионта;

3) заданию, которое было задержано после выполнения;

SJC¤_JOB_BURST -

Это булев код элемента. Он означает, что перед каждым заданием требуется печатать страницу разрыва. Он имеет значение только для выходных очередей выполнения;

SJC¤_NO_JOB_BURST -

Это булев код элемента. Он означает, что не требуется печатать страницу разрыва перед каждым заданием. Он имеет значение только для выходных очередей задания. данный код принят по умолчанию;

SJC$_JOB_COPIES -

Это входной код элемента. Он указывает, сколько раз требуется повторить все печатающее задание. Буфер должен содержать ненулевое целое значение размером в длинное слово. По умолчанию печатающее задание выполняется один раз;

SJC$_JOB_FLAG -

Это булев код элемнента. Он означает, что перед каждым заданием необходимо печатать страницу флагов. Он имеет значение только для выходных очередей выполнения;

SJC$_NO_JOB_FLAG -

Это булев код элемента. Он означает, что не требуется печатать страницу флагов перед каждым заданием. Он имеет значение только для выходных очередей выполнения. Данный код принят по умолчанию;

SJC$_JOB_LIMIT -

Это входной код элемента. Он задает максимальное количество заданий, которые могут одновременно выполняться на одной очереди. Буфер должеч содержать значение размером в длинное слово в диапазоче от 1 до 255. Он имеет значение только для пакетных очередей выполнения. По умолчанию предел заданий равен 1;

SJC$_JOB_NAME -

Это входной код элемента. Он указывает имя задания. Буфер должен содержать строку длиной от 1 до 39 символов. Для кодов функций SJC$_ENTER_FILE, SJC$_CREATE_JOB и SJC$_ALTER_JOB. Код элемента

SJC¤_JOB_NAME определяет имя, которое идентифицирует данное задание. По умолчанию используется имя первого файла в задании. Для кода функции SJC¤_SYNCHRONIZE_JOB, код элемента SJC¤_JOB_NAME определяет имя того задания, которое необходимо синхронизировать. Данное имя задания неявно квалифицируется именем пользователя;

SJC¤_JOB_RESET_MODULES -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Буфер должен содержать имена одного или нескольких текстовых модулей, разделенные запятыми. Данный код элемента означает, что данные модули необходимо извлечь из управляющей библиотеки устройства и копировать перед каждым печатающим заданием;

SJC¤_NO_JOB_RESET_MODULES -

Это булев код элемента. Он означает, что текстовые модули не требуется копировать на ацпу. Данный код принят по умолчанию;

SJC¤_JOB_SIZE_MAXIMUM -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Он означает, что печатающее задачие может выполняться только в том случае, если его общий размер в блоках меньше или равен заданной величине. Буфер определяет данное ненулевое значение размером в длинное слово;

SJC$_NO_JOB_SIZE_MAXIMUM -

Это булев код элемента. Он означает, что печатающее задание может выполняться немедленно, независимо от его размеров. Данный код принят по умолчанию;

SJC$_JOB_SIZE_MINIMUM -

Это входной код элемента. Он имеет значение только для выходных очередей печатающих заданий. Он означает, что печатающее задание может выполняться только в том случае, если его общий размер в блоках больше или равен заданной величине. Буфер определяет данное ненулевое значение размером в длинное слово;

SJC$_NO_JOB_SIZE_MINIMUM -

Это булев код элемента. Он означает, что печатающее задание может выполниться немедленно, независимо от его размера. Данный код принят по умолчанию;

SJC$_JOB_SIZE_SCHEDULING -

Это булев код элемента. Он означает, что задания, которые вводятся в выходную очередь выполнения, необходимо планировать во времени в соответствии с их размером, так, чтобы наименьшее задание с установленным приоритетом обрабатывалось первым. Данный код принят по умолчанию;

SJC$_NO_JOB_SIZE_SCHEDULING -

Это булев код элемента. Он означает, что печатающие задания с установленным приоритетом не требуется планировать в соответствии с их размером. Изменение значения данного кода элемента для очереди в то время,

когда в очереди есть ожидающие печатающие задания, будет давать непредсказуемые результаты;

SJC$_JOB_STATUS_OUTPUT -

Это выходной код элемента. Если он задан, $SNDJBC возвращает в виде символьной строки текстовое сообщение, описывающее состояние подчиненного задания. Поскольку сообщение может включать до 255 символов, то в поле "длина буфера" дескриптора элемента необходимо указывать 255 (байтов);

SJC$_JOB_TRAILER -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он означает, что после каждого задания необходимо печатать дополнительную страницу;

SJC$_NO_JOB_TRAILER -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он означает, что не требуется печатать дополнительную страницу после каждого задания. Данный код принят по умолчанию;

SJC$_LAST_PAGE -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Он задает номер страницы, которой должна заканчиваться печать. Буфер задает данное ненулевое значение размером в длинное слово;

SJC$_NO_LAST_PAGE -

Это булев код элемента. Он означает, что печать должна заканчиваться последней страницей. Данный код принят

по умолчанию;

SJC$_LIBRARY_SPECIFICATION -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Он означает, что управляющей библиотекой устройства является SYS$LIBRARY:имя.TLB, где "имя" - это допустимое имя файла. Буфер должен содержать данное имя файла;

SJC$_NO_LIBRARY_SPECIFICATION -

Это булев код элемента. Он означает, что управляющей библиотекой устройства является SYS$LIBRARY:SYSDEVCTL.TLB. Данный код принят по умолчанию;

SJC$_LOG_DELETE -

Это булев код элемента. Он означает, что файл регистрации, созданный для пакетного задания, необходимо удалить. Он имеет значение только для пакетных заданий. Данный код принят по умолчанию;

SJC$_NO_LOG_DELETE -

Это булев код элемента. Он означает, что файл регистрации, созданный для пакетного задания, не требуется удалять;

SJC$_LOG_QUEUE -

Это входной код элемента. Он имеет значение только для пакетных заданий. Он задает очередь, в которую вводится файл регистрации, созданный для пакетного задания. Буфер должен содержать имя данной очереди. По умолчанию файл регистрации вводится в очередь SYS$PRINT.

Строка имени может содержать строчные или прописные символы (строчные преобразуются в прописные), цифровые символы, знаки денежной единицы и подчеркивания. Если строка является логическим именем, то SYS$SNDJBC итеративно транслирует ее до тех пор, пока не будет найдена эквивалентная строка, или пока количество выполненных трансляций не достигнет предела, допустимого в системе. Максимальная длина окончательной символьной строки - 31 символ; пробелы, знаки табуляции и пустые символы игнорируются;

SJC$_LOG_SPECIFICATION -

Это входной код элемента. Он имеет значение только для пакетных заданий. Он задает спецификацию для файла регистрации, который создается для пакетного задания. Буфер должен содержать спецификацию файла. Поля, опущенные в данной спецификации файла, обеспечиваются из принятой по умолчанию спецификации файла SYS$LOGIN:имя.LOG, где "имя" - это имя задания. Если файл регистрации создается по умолчанию, то для генерации его спецификации используется данная принятая по умолчанию спецификаци файла;

SJC$_NO_LOG_SPECIFICATION -

Это булев код элемента. Он означает, что для созданного пакетного задания файл регистрации создавать не требуется;

SJC$_LOG_SPOOL -

Это булев код элемента. Он означает, что файл регист-

рации, созданный для пакетного задания, требуется печатать. Он имеет значение только для пакетных заданий. Данный код принят по умолчанию;

SJC$_NO_LOG_SPOOL -

Это булев код элемента. Он означает, что файл регистрации пакетного задания не требуется печатать.

SJC$_LOWERCASE -

Это булев код элемента. Он означает, что задание можно выполнять только на том устройстве, которое имеет связанную с устройством характеристику LOWERCASE (строчные символы). Он имеет значение только для выходых очередей выполнения;

SJC$_NO_LOWERCASE -

Это булев код элемента. Он означает, что задание может выполняться независимо от того, имеет или нет выходное устройство характеристику LOWERCASE (строчные символы). Данный код принят по умолчанию;

SJC$_NEW_VERSION -

Это булев код элемента. Он означает, что необходимо создать новую версию системного файла очереди заданий, независимо от того, существует данный файл или нет. По умолчанию системный файл очереди заданий создается в том случае, если он еще не судествует;

SJC$_NEXT_JOB -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он означает, что текущее задание требуется прерывать, а печать возобновлять со

следующего задания;

SJC$_NOTE -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Он задает строку, которую требуется напечатать на страницах флагов задания и флагов файла. Буфер должен содержать данную строку;

SJC$_NO_NOTE -

Это булев код элемента. Он означает, что никакую строку не требуется печатать на страницах флагов задания и флагов файла. Данный код принят по умолчанию;

SJC$_NOTIFY -

Это булев код элемента. Он означает, что по завершении задания необходимо на все подключенные терминалы разослать сообщение от имени того пользователя, которому подчинено задание;

SJC$_NO_NOTIFY -

Это булев код элемента. Он означает, что по завершении задания не требуется рассылать сообщение. Данный код принят по умолчанию;

SJC$_OPERATOR_REQEST -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Буфер должен содержать строку текста. Данный код означает, что когда задание начинает выполняться, очередь выполнения необходимо остановить, а заданную строку текста требуется включить в сообщение, предназначенное для программы обслуживания запросов к оператору очереди;

SJC¤_NO_OPERATOR_REQEST -

Это булев код элемента. Он означает, что входную очередь выполнения не требуется останавливать, и никакое сообщение не тербуется посылать оператору очереди. Данный код принят по умолчанию;

SJC¤_OWNER_UIC -

Это входной код элемента. Он задает UIC владельца очереди. Буфер должен содержать UIC размером в длинное слово. По умолчанию UIC владельца - [1,4];

SJC¤_PAGE_HEADER -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он означает, что заголовок страницы необходимо печатать на каждой выводимой странице;

SJC¤_NO_PAGE_HEADER -

Это булев код элемента. Он означает, что не требуется печатать заголовок страницы. Данный код принят по умолчанию;

SJC¤_PAGE_SETUP_MODULES -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Буфер должен содержать одно или несколько имен текстовых модулей, разделенных запятыми. Данный код означает, что перед печатью каждой страницы необходимо извлечь данные модули из управляющей библиотеки устройства и скопировать их на АЦПУ;

SJC$_NO_PAGE_SETUP_MODULES -

Это булев код элемента. Он означает, что управляющие модули устройства не надо копировать. Данный код принят по умолчанию;

SJC$_PAGINATE -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он означает, что симбионт должен делить выводимую информацию на страницы, вставляя продвижение формы всякий раз, когда выводимая информация достигает нижнего поля формы. Данный код принят по умолчанию;

SJC$_NO_PAGINATE -

Это булев код элемента. Он означает, что симбионт не должен делить на страницы выводимую информацию;

код от SJC$_PARAMETER_1 до SJC$_PARAMETER_8 -

Это входные коды элемента; их восемь, последняя цифра кода элемента может изменяться от 1 до 8. Для каждого заданного кода элемента буфер должен содержать строку длиной не более 255 символов. Для пакетных заданий данная строка становится значением параметра соответственно от P1 до P8 внутри самой внешней командной процедуры. Для печатающих заданий система делает данную строку доступной симбионту, хотя стандартные печатающие симбионты МОС ЭП не используют данную информацию. По умолчанию каждый из восьми парамеров задает пустую строку. Если какой-либо элемент SJC$_PARAMETER задан для кода функции SJC$_ALTER_JOB, то значением

каждого незаданного элемента является пустая строка;

SJC$_NO_PARAMETRES -

Это булев код элемента. Он означает, что элементы SJC$_PARAMETER не передаются пакетному или печатающему заданию. Данный код принят по умолчанию;

SJC$_PASSALL -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он означает, что симбионт должен печатать файл в режиме PASSALL;

SJC$_NO_PASSALL -

Это булев код элемента. Он означает, что симбионт не должен печатать файл в режиме PASSALL. Данный код принят по умолчанию;

SJC$_PRIORITY -

Это входной код элемента. Буфер должен задавать значение размером в длинное слово в диапазоне от 0 до 255. Данное значение определяет диспетчерский приоритет задания в очереди относительно диспетчерских приоритетов других заданий в той же очереди. По умолчанию диспетчерским приоритетом задания является значение параметра системной генерации DEFQUEPRI. Если значение DEFQUEPRI равно 0, то принятым по умолчанию диспетчерским приоритетом является базовый приоритет запрашивающего процесса. Если значение, указанное для SJC$_PRIORITY больше, чем параметр системной генерации MAXQUEPRI, то используется значение MAXQUEPRI;

SJC$_PROCESSOR -

Это входной код элемента. Буфер должен задавать допустимое имя файла. Если код SJC$_PRCESSOR задан для выходной очереди выполнения, то он означает, что необходимо выполнить образ симбионта SYS$SYSTEM:имя.EXE, где "имя" - это имя файла, содержащееся в буфере. Если код SJC$_PROCESSOR задан для общей выходной очереди, то он означает, что общая очередь может помещать задания только в служебные очереди, которые выполняют образ симбионта SYS$SYSTEM:имя.EXE, где ""имя" - это имя файла, содержащееся в буфере;

SJC$_NO_PROCESSOR -

Это булев код элемента. Он означает, что должен выполняться образ симбионта SYS$SYSTEM:PRTSMB.EXE. Данный код принят по умолчанию;

SJC$_PROTECTION -

Это входной код элемента. Он определяет защиту очереди. Буфер должен содержать длинное слово в формате, показанном на рис.81.

Маска защиты очереди

Рис. 81

Биты с 0 по 15 задают значение защиты - четыре типа доступа: чтение (R), запись (W), выполнение (E), удаление (D), которые должны предоставляться четырем классам пользователей (система, владалец, группа, все). Установленные в 1 биты запрещают доступ, а очищенные биты позволяют доступ.

Биты с 16 по 31 включают или выключают биты с 0 по 15. Если какой-то бит во втором слове установлен в 1, то соответствующий бит в первом слове будет влиять на защиту очереди. Если бит во втором слове очищен, то соответствующий бит в первом слове игнорируется.

По умолчанию защита очереди - (S:E, O:D, G:R, W:W);

SJCB_QUEUE -

Это входной код элемента. Он задает очередь, на которую направлена данная операция. Буфер должен содержать

имя очереди. Строка может содержать прописные или строчные символы (строчные преобразуются в прописные), цифровые символы, знаки денежной единицы и подчеркивания. Если строка является логическим именем, то SYS¤SNDJBC итеративно транслирует ее до тех пор, пока не будет найдена эквивалентная строка или пока количество выполненных трансляций не достигнет предела, допустимого в системе. Максимальная длина окончательной символьной строки - 31 символ; пробелы, знаки табуляции и пустые символы игнорируются;

SJC¤_QUEUE_FILE_SPECIFICATION -

Это входной код элемента. Он определяет спецификацию файла для системного файла очереди заданий. Буфер должен содержать допустимую спецификацию файла. Поля, опущенные в данной спецификации файла, берутся из принятой по умолчанию спецификации файла SYS¤SYSTEM:JBCSYSQUE.DAT;

SJC¤_RECORD_BLOCKING -

Это булев код элемента. Он имеет значение только для выходных очередей выполнения. Он означает, что симбионт может сцеплять выходные записи, которые он посылает на выходное устройство. Для стандартных симбионтов операционная система МОС ВП блокирование записей может дать значительные преимущества в производительности по сравнению с режимом единичных записей. Данный код принят по умолчанию;

SJC¤_NO_RECORD_BLOCKING -

Это булев код элемента. Он означает, что симбионт должен посылать каждую запись на выходное устройство в отдельном запросе ввода-вывода;

SJC¤_RELATIVE_PAGE -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Буфер должен содержать целое значение со знаком размером в длинное слово. Данный код элемента означает, что печать необходимо возобновить после пропуска вперед (если в буфере положительное значение) или назад (если в буфере отрицательное значение) заданного количества страниц;

SJC¤_REQUEUE -

Это булев код элемента. Он означает, что задание необходимо снова поставить в очередь. По умолчанию задание удаляется;

SJC¤_RESTART -

Это булев код элемента. Он означает, что задание можно снова запускать после системного сбоя или снова ставить в очередь после выполнения. Данный код принят по умолчанию для печатающих заданий;

SJC¤_NO_RESTART -

Это булев код элемента. Он означает, что задание нельзя выполнять после системного сбоя или после повторной постановки в очередь. Данный код принят по умолчанию для пакетных заданий;

SJC$_RETAIN_ALL_JOBS -

Это булев код элемента. Он означает, что задания после их выполнения необходимо сохранять в очереди вместе с кодом завершения;

SJC$_RETAIN_ERROR_JOBS -

Это булев код элемента. Он означает, что необходимо сохранять только те задания, которые выполнялись неуспешно (в коде завершения задания младший бит очищен);

SJC$_NO_RETAIN_JOBS -

Это булев код элемента. Он означает, что задания после их завершения не требуется оставлять в очереди. Данный код принят по умолчанию;

SJC$_SEARCH_STRING -

Это входной код элемента. Он имеет значение только для выходных очередей выполнения. Данный код элемента означает, что печать необходимо возобновить со страницы, на которой впервые появляется заданная строка. Поиск данной строки осуществляется в направлении вперед;

SJC$_SWAP -

Это булев код элемента. Он имеет значение только для пакетных очередей выполнения. Он означает, что задание, инициированное из очереди, можно выгружать. Данный код принят по умолчанию;

SJC$_NO_SWAP -

Это булев код элемента. Он означает, что задание нельзя выгружать;

SJC$_TERMINAL -

Это булев код элемента. Он имеет значение только для выходных очередей. Он определяет тип очереди, как очередь, направляемую на терминал вместо ацпу;

SJC$_NO_TERMINAL -

Это булев код элемента. Он определяет очередь, как очередь, направляемую на ацпу вместо терминала;

SJC$_TOP_OF_LINE -

Это булев код элемента. Он имеет значение только для выходных очередей. Он означает, что печать необходимо возобновить с начала файла;

SJC$_UIC -

Это входной код элемента. Он задает 4-байтовый код идентификации (UIC) того пользователя, для которого делается данный запрос. По умолчанию имя пользователя берут из запрашивающего процесса. Данный код элемента требует привилегии CMKRNL;

SJC$_WSDEFAULT -

Это входной код элемента. Он имеет значение только для пакетных заданий и для очередей выполнения. Он определяет принимаемый по умолчанию размер рабочего набора для пакетных заданий или заданий, инициированных из пакетной очереди, либо принимаемый по умолчанию размер рабочего набора симбионтного процесса, связанного с выходной очередью. Симбионтный процесс может управлять несколькими выходными очередями; однако, принимаемый по умолчанию размер рабочего набора для симбионтного

процесса устанавливается первой очередью, к которой он подсоединяется. Буфер должен содержать целое значение от 1 до 65535;

SJC$_NO_WSDEFAULT -

Это булев код элемента. Он означает, что значения, принимаемые по умолчанию для рабочего набора, определяет система. Данный код принят по умолчанию. Для пакетных заданий принимаемый по умолчанию размер рабочего набора и экстент рабочего набора (максимальный размер) включаются в каждую пользовательскую запись в системном файле авторизации пользователя (UAF). Значения для данных элементов можно задавать для отдельных заданий и/или для всех заданий данной очереди. В табл.70 показано, какие действия предпринимаются, если для SJC$_WSDEFAULT задано какое-либо значение;

Таблица 70

Задано значение для задания?	Задано значение для очереди?	Предпринимаемые действия
нет	нет	используется значение UAF
нет	да	используется значение для очереди
да	да	используется меньшее из двух значений
да	нет	сравнивается заданное значение со значением UAF; используется мень-

Продолжение табл. 70

Задано значение для задания?	Задано значение для очереди?	Предпринимаемые действия
	шее	

SJC$_WSEXTENT -

Это входной код элемента. Он имеет значение только для пакетных заданий и очередей выполнения. Он задает экстент рабочего набора для пакетных заданий или заданий, инициированных из пакетной очереди, либо экстент рабочего набора для симбионтного процесса, связанного с выходной очередью. Симбионтный процесс может управлять несколькими выходными очередями; однако, экстент рабочего набора для симбионтного процесса устанавливается первой очередью, к которой он подсоединяется. Буфер должен содержать целое значение размером в длинное слово в диапазоне от 1 до 65535;

SJC$_NO_WSEXTENT -

Это булев код элемента. Он означает, что экстент рабочего набора определяет система. Данный код принят по умолчанию. Действия, которые предпринимаются, если для пакетного задания указано некоторое значение SJC$_WSEXTENT, описаны при обсуждении кода элемента SJC$_WSDEFAULT и в табл.70;

SJC$_WSQUOTA -

Это входной код элемента. Он имеет значение только для пакетных заданий и очередей выполнения. Он задает квоту рабочего набора для пакетных заданий или заданий, инициированных из пакетной очереди, либо квоту рабочего набора для симбионтного процесса, связанного с выходной очередью. Симбионтный процесс может управлять несколькими выходными очередями; однако, квота рабочего набора для симбионтного процесса устанавливается первой очередью, к которой он подсоединяется. Буфер должен содержать целое значение размером в длинное слово в диапазоне от 1 до 65535;

SJC$_NO_WSQUOTA -

Это булев код элемента. Он означает, что квоту рабочего набора определяет система. Данный код принят по умолчанию.

Действия, которые предпринимаются, если для пакетного задания указано некоторое значение SJC$_WSQUOTA, описаны при обсуждении кода элемента SJC$_WSDEFAULT и в табл.70;

IOSB

Использование в МОС ВП: IO_STATUS_BLOCK

Тип: квадрослово (без знака)

Доступ: только запись

Механизм: по ссылке

Блок состояния ввода-вывода, в который $SNDJBC записывает код завершения после окончания запрошенной операции. Аргумент IOSB - это адрес блока состояния ввода-вывода.

При инициации запроса ¤SNDJBC записывает 0 в блок состояния ввода-вывода размером в квадрослово. По завершении запрошенной операции ¤SNDJBC записывает код состояния в первое длинное слово блока состояния ввода-вывода. Во второе длинное слово ¤SNDJBC записывает 0; данное длинное слово не используется.

Коды состояния, которые ¤SNDJBC возвращает в блок состояния ввода-вывода, - это обычно коды состояния из диспетчера заданий. Данные коды состояния определяют при помощи макрокоманды ¤JBCMSGDEF. В некоторых случаях ¤SNDJBC может возвращать коды состояния, полученные в результате ошибки в программе системного обслуживания или в программе СУД-32, которые используются при выполнении данного запроса. Для запроса SJC¤_SYNCHRONIZE_JOB возвращается код завершения запрошенного задания.

Хотя данный аргумент не является обязательным, рекомендуется указывать его по следующим причинам:

1) если для сигнализации о завершении программы системного обслуживания используется флаг события, то код состояния в блоке состояния ввода-вывода можно проверить, чтобы убедиться в том, что данный флаг не был установлен событием, отличным от завершения программы системного обслуживания;

2) если для синхронизации завершения программы системного обслуживания используется программа системного обслуживания ¤SYNCH, то блок состояния ввода-вывода является обязательным аргументом для ¤SYNCH;

3) код состояния, возвращаемый в R0, и код состояния возвращаемый в блок состояния ввода-вывода, обеспечивают информацию о разных аспектах вызова программы системного обслуживания $SNDJBC. Код состояния, возвращаемый в R0, дает информацию о том, насколько успешно осуществился сам вызов программы системного обслуживания; код состояния, возвращаемый в блок состояния ввода-вывода, дает информацию о том, насколько успешно отработала программа системного обслуживания. Следовательно, для того, чтобы точно оценить, насколько успешным был вызов $SNDJBC, необходимо проверить коды состояния, возвращаемые как в R0, так и в блок состояния ввода-вывода.

ASTADR

Использование в MOC ВП: AST_PROCEDURE

Тип: маска входа программы

Доступ: вызов без развертывания стека

Механизм: по ссылке

Подпрограмма обработки AST, которую необходимо выполнить по завершении $SNDJBC. Аргумент ASTADR - это адрес маски входа данной подпрограммы.

Подпрограмма AST, если она задана, выполняется в том же режиме доступа, что и процесс, вызывающий $SNDJBC.

ASTPRM

Использование в MOC ВП: USER_ARG

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Параметр AST, который необходимо передать подпрограмме обработки AST, заданной аргументом ASTADR. Аргумент ASTPRM - это параметр размером в длинное слово.

Описание:

Типы очередей

Операционная система MOC ЭП поддерживает несколько типов очередей, которые помогают в обработке пакетных и печатающих заданий. Различные типы очередей можно разделить на три основные категории в соответствии со способом, которым система обрабатывает задания, назначенные в данную очередь. Существует три типа очередей - очередь выполнения, общая и логическая очередь. Очередь выполнения планирует задания для выполнения; общие и логические очереди передают задания в очереди выполнения. Внутри данных трех категорий тип очереди далее определяется видами заданий, которые данная очередь может принимать для обработки. Некоторые типы общей очереди и очереди выполнения допускают пакетные задания; другие типы допускают печатающие задания. Логические очереди ограничиваются печатающими заданиями.

Очередь создается при помощи вызова $SNDJBC с указанием кода функции SJC$_CREATE_QUEUE. Коды элемента, которые заданы в данном вызове, определяют тип создаваемой очереди.

Очередь выполнения планирует задания для обработки. Существует два типа очереди выполнения: пакетная очередь выполнения и выходная очередь выполнения.

Пакетная очередь выполнения может планировать работу только пакетных заданий. Пакетное задание работает как

отсоединенный процесс, который последовательно выполняет одну или несколько командных процедур; список командных процедур определяют как часть начального описания задания. Пакетную очередь выполнения создают, задавая код элемента SJC$_BATCH в вызове программы системного обслуживания $SNDJBC.

Выходная очередь выполнения планирует печатающие задания для обработки их независимым симбионтным процессом, который связан с данной очередью. Диспетчер заданий посылает данному симбионту список файлов, которые необходимо обработать; данный список файлов определяют как часть начального описания задания. По мере того, как симбионт обрабатывает каждый файл, он формирует выходную информацию для устройства, которым он управляет, такое как АЦПУ или терминал.

Стандартный образ печатающего симбионта, обеспечиваемый операционной системой МОС ВП, предназначен для печати файлов. Кроме того, могут быть симбионты, модифицированные или написанные пользователем, которые предназначены для той или иной деятельности по обработке файлов под управлением средств пакетной обработки. Образ симбионта, который обрабатывает задания из выходной очереди, задают при помощи кода элемента SJC$_PROCESSOR. Если данный код элемента опущен, то с очередью связывается стандартный образ печатающего симбионта МОС ВП, PRTSMB.

Существует три типа выходной очереди выполнения:

1) очередь выполнения АЦПУ. Данный тип очереди обычно использует стандартный печатающий симбионт для того, чтобы направить выходную информацию на АЦПУ. В коде элемента SJC¤_PROCESSOR можно задавать симбионт, обеспеченный пользователем. Очередь выполнения АЦПУ - это принимаемый по умолчанию тип выходной очереди выполнения;

2) очередь выполнения терминала. Данный тип очереди обычно использует стандартный печатающий симбионт для того, чтобы направить выходную информацию на терминал. В коде элемента SJC¤_PROCESSOR можно задавать симбионт, обеспеченный пользователем. Очередь выполнения терминала задают при помощи кода элемента SJC¤_TERMINAL во время создания выходной очереди выполнения;

3) служебная очередь выполнения. Данный тип очереди используется для обработки файлов, принадлежащих заданиям в очереди, модифицированным или написанным пользователем симбионтом, который указан в коде элемента SJC¤_PROCESSOR. Выходную очередь выполнения нельзя пометить при ее создании, как служебную очередь выполнения. Только обеспеченный пользователем симбионт может указать, что очередь, которая с ним связана, должна быть помечена как служебная очередь выполнения.

Следовательно, при создании выходной очереди выполнения ее можно изначально пометить как очередь выполнения либо АЦПУ, либо терминала. Однако, когда очередь уже запущена, симбионтный процесс, связанный с данной очередью,

может изменить тот тип очереди, который был назначен при ее создании, на очередь выполнения ацпу, терминала или служебную очередь выполнения:

1) если запускается выходная очередь выполнения, которая связана со стандартным печатающим симбионтом МОС ЭП, симбионт определяет, управляет он ацпу или терминалом. Он сообщает данную информацию диспетчеру заданий. Затем диспетчер заданий изменяет, если необходимо, тип, назначенный выходной очереди выполнения;

Если запускается выходная очередь выполнения, которая связана с модифицированным или написанным пользователем симбионтом, симбионт может идентифицировать данную очередь, как служебную, для диспетчера заданий. Если написанный или модифицированный пользователем симбионт не извещает диспетчера заданий о том, что он хочет изменить обозначение типа очереди, то выходной очереди выполнения будет оставлено то обозначение типа очереди, которое она получила при создании. То есть, она остается помеченной, как очередь выполнения ацпу или терминала, несмотря на то, что она связана с нестандартным симбионтом.

Общая очередь сохраняет задания до тех пор, пока не появится соответствующая очередь выполнения, способная инициировать данное задание, тогда диспетчер заданий переставляет данное задание в имеющуюся очередь выполнения.

Общую очередь создают при помощи кода элемента SJC$_GENERIC_QUEUE, заданного в вызове программы системного обслуживания $SNDJBC. Пользователь определяет при помощи

кода элемента SJC¤_GENERIC_TARGET каждую очередь выполнения, в которую данная общая очередь может направлять задания. Поскольку общая очередь может направлять задания в несколько очередей выполнения, то пользователь может указывать код элемента SJC¤_GENERIC_TARGET в одном вызове ¤SNDJBC до 124 раз, чтобы определить полный набор очередей выполнения для какой-либо общей очереди. Если код элемента SJC¤_GENERIC_TARGET не указан, то общая очередь направляет задания в любую очередь выполнения, которая имеет тот же тип очереди, что и общая очередь: то есть, общая пакетная очередь будет направлять задание в любую доступную пакетную очередь выполнения, и т.д. Имеется одно исключение: общая очередь не будет направлять задание в ту очередь выполнения, которая была создана с указанием кода элемента SJC¤_NO_GENERIC_SELECTION в вызове ¤SNDJBC.

Существует два типа общей очереди: общая пакетная очередь и общая выходная очередь.

Общая пакетная очередь может направлять задания только в пакетные очереди выполнения. Общую пакетную очередь создают, задавая в вызове программы системного обслуживания ¤SNDJBC как код элемента SJC¤_GENERIC_QUEUE, так и SJC¤_BATCH.

Общая выходная очередь может направлять задания к любому из трех типов выходной очереди выполнения: очереди ацпу, терминала или служебной очереди. Можно создать общую выходную очередь, которая направляет задания к любой комбинации данных трех типов выходной очереди выполнения. Одна-

ко, обычно при создании выходной очереди задают список очередей об'ектов конкретного типа. Таким образом, такая общая выходная очередь направляет задания к одному типу выходной очереди выполнения. Следовательно, можно управлять тем, будут ли задания, подчиненые общей выходной очереди выполнения выводить информацию на ацпу или будут посылать информацию для обработки служебному симбионту. Общую выходную очередь создают, задавая код элемента SJC$_GENERIC_QUEUE.

Логическая очередь выполняет ту же функцию, что и общая выходная очередь, за исключением того, что логическая очередь может направлять задания только в одну очередь выполнения ацпу, терминала или в служебную очередь. Логическая очередь - это выходная очередь, которой назначено передавать ее задания в одну из очередей выполнения.

Для того, чтобы сделать какую-либо выходную очередь логической, необходимо вызвать программу системного обслуживания $SNDJBC, при этом данная выходная очередь должна быть остановлена. Данный вызов должен указывать код функции SJC$_ASSIGN_QUEUE и код элемента SJC$_DESTINATION_QUEUE. Код элемента SJC$_DESTINATION_QUEUE используют для того, чтобы указать очередь выполнения, в которую логическая очередь должна направлять задания. Когда логическая очередь запускается, она автоматически переставляет свои задания в указанную очередь выполнения, как только данная очередь выполнения становиться доступной. Логическую очередь можно вернуть к ее исходному определению выходной очереди, задавая код функции SJC$_DEASSIGN_QUEUE в последующем вызове

программы системного обслуживания ¤SNDJBC.

Защита очереди и задания

Существуют три аспекта защиты очереди:

1) очередь имеет соответствующий UIC. При создании очереди ей назначается UIC при помощи кода элемента SCJ¤_OWNER_UIC. Если данный код элемента не задан, то очередь получает принятый по умолчанию UIC;

2) очереди можно присвоить маску защиты, задавая код элемента SCJ¤_PROTECTION. Данная маска защиты определяет доступ на чтение, запись, выполнение и удаление для четырех категорий пользователя: владелец, группа, система и все;

3) некоторые операции, связанные с очередями, требуют наличия определенных привилегий у процесса, вызывающего ¤SNDJBC.

Защита задания осуществляется единственным способом. Если задание подчинено очереди, то ему присваивается такой же UIC, как UIC процесса, которому подчинено данное задание, за исключением того случая, когда указан код элемента SJC¤_UIC, задающий другой UIC.

Для каждой запрошенной операции программа системного обслуживания ¤SNDJBC сравнивает UIC и привилегии запрашивающего процесса с UIC очереди, с защитой, указанной для очереди, и с привилегией (привилегиями), если для данной операции требуются какие-либо привилегии. Данная проверка осуществляется таким же способом, каким проверяется возможность доступа к файлу, сравнивая UIC владельца и защиту файла с UIC и привилегиями запрашивающего процесса.

Для операций, которые осуществляются применительно к заданиям, проверяется защита "R" (чтение) и "D" (удаление), указанная для той очереди, в которую введено данное задание, и UIC владельца задания. Вообще доступ "R" к заданию дает пользователю возможность определить, что данное задание существует; доступ "D" к заданию дает пользователю возможность воздействовать на данное задание.

Для операций, которые осуществляются применительно к очередям, проверяется защита "W" (запись) и "E" (выполнение), указанная для данной очереди, и UIC владельца очереди. В общем случае доступ "W" к очереди дает пользователю возможность подчинять задания данной очереди; доступ "E" к очереди дает пользователю возможность поступать как оператор очереди, включая способность воздействовать на задания в данной очереди, на ведение учетной информации, а также изменять очереди. Привилегия OPER предоставляет доступ "E" ко всем очередям.

Привилегии и ограничения

Для того, чтобы задать следующие коды функции, вызывающий процесс должен иметь одновременно привилегии OPER и SYSNAM:

1) SJC$_START_QUEUE_MANAGER;

2) SJC$_STOP_QUEUE_MANAGER.

Для того, чтобы задать следующие коды функции, вызывающий процесс должен иметь привилегию OPER:

1) SJC$_CREATE_QUEUE;

2) SJC$_DEFINE_CHARACTERISTIC;

3) SJC¤_DEFINE_FORM;

4) SJC¤_DELETE_CHARACTERISTIC;

5) SJC¤_DELETE_FORM;

6) SJC¤_DELETE_QUEUE;

7) SJC¤_START_ACCOUNTING;

8) SJC¤_STOP_ACCOUNTING.

Для того, чтобы задать следующий код функции, вызывающий процесс должен иметь привилегию OPER, доступ "E" к очереди, которая содержит указанное задание или доступ "R" к указанному заданию:

SJC¤_SYNCHRONIZE_JOB.

Для того, чтобы задать следующие коды функций, вызывающий процесс должен иметь привилегию OPER, доступ "E" к указанной очереди или доступ "W" к указанной очереди:

1) SJC¤_CREATE_JOB;

2) SJC¤_ENTER_FILE.

Для того, чтобы задать следующие коды функций, вызывающий процесс должен иметь привилегию OPER, доступ "E" к указанной очереди (очередям):

1) SJC¤_ALTER_QUEUE;

2) SJC¤_ASSIGN_QUEUE;

3) SJC¤_DEASSIGN_QUEUE;

4) SJC¤_MERGE_QUEUE;

5) SJC¤_PAUSE_QUEUE;

6) SJC¤_RESET_QUEUE;

7) SJC¤_START_QUEUE;

8) SJC¤_STOP_QUEUE.

Для того, чтобы задать следующие коды функции, вызывающий процесс должен иметь привилегию OPER, доступ "E" к очереди, которая содержит указанное задание или доступ "D" (удаление) к указанному заданию:

1) SJC$_ABORT_JOB;

2) SJC$_ALTER_JOB;

3) SJC$_DELETE_JOB.

Для того, чтобы задать следующие коды функции, никакие привилегии не требуются:

1) SJC$_ADD_FILE;

2) SJC$_BATCH_CHECKPOINT;

3) SJC$_CLOSE_DELETE;

4) SJC$_CLOSE_JOB;

5) SJC$_WRITE_ACCOUNTING.

Для того, чтобы задать базовый приоритет (при помощи кода элемента SJC$_BASE_PRIORITY) более высокий, чем базовый приоритет запрашивающего процесса, требуется привилегия OPER или ALTPRI.

Для того, чтобы задать базовый приоритет (при помощи кода элемента SJC$_BASE_PRIORITY) более высокий, чем значение параметра системной генерации DEFQUEPRI, требуется привилегия OPER или ALTPRI. Наибольшее значение, которое можно задавать, определяется параметром системной генерации MAXQUEPRI.

Для того, чтобы задать следующие коды элемента, вызывающий процесс должен иметь привилегию OPER:

1) SJC¤_PROTECTION;

2) SJC¤_OWNER_UIC.

Для того, чтобы задать следующие коды элемента, вызывающий процесс должен иметь привилегию CMKRNL:

1) SJC¤_ACCOUNT_NAME;

2) SJC¤_UIC;

3) SJC¤_USERNAME.

Возвращаемые значения кода состояния:

SS¤_NORMAL - успешное завершение;

SS¤_ACCVIO - вызывающий процесс не может прочитать список элементов или входной буфер, либо не может записать буфер возвращаемой длины, выходной буфер или блок состояния;

SS¤_BADPARAM - недопустимый код функции; список элементов содержит недопустимый код элемента; дескриптор буфера имеет недопустимую длину или зарезервированный параметр имеет ненулевое значение;

SS¤_DEVOFFLINE - процесс диспетчера заданий не работает;

SS¤_EXASTLM - задан аргумент ASTADR, и процесс превысил свою квоту ASTLM;

SS¤_ILLEFC - аргумент EFN задает недопустимый номер флага события;

SS¤_INSFMEM - в стеке режима управления не хватает места для выполнения данного запроса;

SS$_MBFULL - почтовый ящик диспетчера заданий переполнен;

SS$_MBTOOSML - сообщение, посылаемое в почтовый ящик, слишком велико для почтового ящика диспетчера заданий;

SS$_UNASEFC - аргумент EFN задает кластер флагов событий, с которым процесс не соединен.

Коды состояния, возвращаемые в блок состояния ввода-вывода:

JBC$_NORMAL - успешное завершение;

JBC$_DELACCESS - защита указанного файла, который был введен с операцией "удаление", запрещает доступ на удаление для данного вызывающего процесса;

JBC$_DUPFORM - заданный номер формы недопустим, потому что он уже определен; каждая форма должна иметь уникальный номер;

JBC$_EMPTYJOB - открытое задание нельзя закрыть, потому что оно не содежит файлов;

JBC$_EXECUTING - параметры указанного задания нельзя модифицировать, потому что задание уже выполняется;

JBC$_INCDSTQUE - тип очереди с указанным обозначением несовместим с запрошенной операцией;

JBC$_INCFORMPAR - указанные параметы длины и ширины, формы и размеры полей несовместимы;

значение разности размеров верхнего и нижнего поля должно быть меньше, чем длина формы, разность размеров параметров левого и правого поля должна быть меньше, чем ширина строки;

JBC¤_INCOMPLETE - запрошенную операцию управления очередью нельзя выполнить, поскольку еще не завершилась ранее запрошенная операция управления очередью;

JBC¤_INCQUETYP - тип указанной операции несовместим с запрошенной операцией;

JBC¤_INVCHANAM - заданное имя характеристики синтаксически неверно;

JBC¤_INVDSTQUE - имя очереди назначения синтаксически неверно;

JBC¤_INVFORNAM - имя формы синтаксически неверно;

JBC¤_INVFUNCOD - указанный код функции недопустим;

JBC¤_INVITMCOD - список элементов содержит недопустимый код элемента;

JBC¤_INVPARLEN - длина указанной строки превышает допустимый диапазон для данного кода элемента;

JBC¤_INVPARVAL - значение параметра, заданного для кода элемента, выходит за диапазон, допустимый для данного кода элемента;

JBC¤_INVQUENAM - имя очереди синтаксически неверно;

JBC¤_JOBQUEDIS - запрос нельзя выполнить, потому что

не запущен системный планировщик очереди заданий;

JBC¤_MISREQPAR - не задан код элемента, который является обязательным для указанного кода функции;

JBC¤_NODSTQUE - указанная очередь назначения не существует;

JBC¤_NOOPENJOB - запрашивающий процесс не открывал задание с функцией SJC¤_CREATE_JOB;

JBC¤_NOPRIV - защита очереди запрещает доступ к очереди для заданной операции;

JBC¤_NOQUESPACE - системный файл очереди заданий переполнен, и его нельзя расширить;

JBC¤_NORESART - указанное задание нельзя снова поставить в очередь, потому что оно не было определено, как повторно запускаемое;

JBC¤_NOSUCHCHAR - указанная характеристика не существует;

JBC¤_NOSUCHFORM - указанная форма не существует;

JBC¤_NOSUCHJOB - указанное задание не существует;

JBC¤_NOSUCHQUE - указанная очередь не существует;

JBC¤_NOTASSIGN - нельзя отменить назначение указанной очереди, потому что она не назначена;

JBC¤_QUENOTSTOP - указанную очередь нельзя отменить, потому то она не остановлена;

JBC¤_REFERENCED - указанную очердь нельзя отменить,

потому что существуют обращения из других очередей или заданий;

JBC¤_STARTED указанную очередь нельзя запустить, потому что она уже работает.

13.107. ¤SNDJBCW - сообщить диспетчеру заданий и ждать завершения

Программы системного обслуживания ¤SNDJBCW и ¤GETQUI в совокупности обеспечивают пользователю интерфейс со средством "диспетчер задания" (JBC). Программа системного обслуживания ¤SNDJBC позволяет создавать и останавливать очереди, а также управлять очередями и заданиями в данных очередях.

Программа системного обслуживания ¤SNDJBCW ставит запрос в очередь к диспетчеру заданий. Для большинства операций ¤SNDJBCW завершается синхронно, то есть, возвращает управление вызывающему процессу после завершения операции. Однако, если запрашивающая операция заключается в том, чтобы приостановить очередь, остановить очередь или прервать работу задания, то ¤SNDJBCW возвращает управление вызывающему процессу после постановки запроса в очередь. Не существует средств для синхронизации завершения данных операций. Кроме того, ¤SNDJBCW не ждет завершения задания, прежде чем вернуть управление вызывающему процессу; для того, чтобы синхронизировать завершение задания, вызывающий процесс должен указать код функции SJC¤SYNCHRONIZE_JOB.

Программа системного обслуживания $SNDJBCW идентична программе "сообщить диспетчеру заданий" ($SNDJBC) за исключением того, что $SNDJBCW завершается синхронно; программа системного обслуживания $SNDJBC возвращает управление вызывающему процессу после завершения постановки запроса в очередь, не ожидая завершения операции.

Дополнительная информация о программе системного обслуживания $SNDJBCW содержится в описании программы системного обслуживания $SNDJBC.

Формат:

SYS$SNDJBCW [EFN],FUNC,NULLARG[,ITMLST][,IOSB]
[,ASTADR][,ASTPRM]

13.108. $SNDOPR - послать сообщение оператору

Программа системного обслуживания $SNDOPR выполняет следующие функции:

1) посылает пользовательский запрос на операторские терминалы;

2) посылает запрос на отмену запроса пользователя на операторские терминалы;

3) посылает ответ оператора на пользовательский терминал;

4) разрешает использовать какой-либо терминал в качестве операторского;

5) показывает состояние операторского терминала;

6) инициализирует операторский файл регистрации.

Формат:

SYS$SNDOPR MSGBUF,[CHAN]

Аргументы:

MSGBUF

Использование в МОС ВП: CHAR_STRING

Тип: строка текста

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки с фиксированной длиной

Пользовательский буфер, задающий операцию, которую необходимо выполнить, и информацию, необходимую для выполнения данной операции. Аргумент MSGBUF - это адрес дескриптора символьной строки, указывающего на буфер.

Формат и содержимое буфера меняются в зависимости от запрошенной операции; однако, первый байт в любом буфере - это код запроса, который задает операцию, подлежащую выполнению.

Макрокоманда $OPCMSG определяет символические имена для данных кодов запроса:

OPC$_RQ_RQST - посылает пользовательский запрос на операторские терминалы. Данный код запроса используется для того, чтобы сделать запрос оператору. Если указано, что на запрос необходимо получить ответ (при помощи аргумента CHAN), то запрос оператору останется активным до тех пор, пока оператор не ответит;

OPC$_RQ_CANCEL - посылает запрос на отмену запроса пользователя на операторские терминалы. Данный код запроса

используется для того, чтобы сообщить одному или нескольким операторам, что предыдущий запрос необходимо отменить. Если задан код OPC$RQ_CANCEL, то аргумент CHAN также должен быть указан;

OPC$_RQ_REPLY - посылает ответ оператора пользователю, который сделал запрос. Данный код запроса используется операторами для отчета о состоянии пользовательского запроса. Форматом буфера сообщения для данного запроса остается формат ответа, найденного в пользовательском почтовом ящике по завершении вызова $SNDOPR. Все функции $SNDOPR, которые доставляют ответ в почтовый ящик, делают это в формате, описанном для данного кода запроса;

OPC$_RQ_TERME - разрешает использовать терминал в качестве операторского терминала. Данный запрос используется для того, чтобы разрешить указанному терминалу получать операторские сообщения;

OPC$_RQ_STATUS - сообщает о состоянии операторского терминала. Данный запрос используется операторами для того, чтобы показать классы операторов, для которых доступен указанный терминал, и список выставленных запросов;

OPC$_RQ_LOGI - инициализирует операторский файл регистрации.

На рис.82, рис.83, рис.84, рис.85, рис.86, рис.87 описан буфер сообщения для каждого из перечисленных кодов запроса. Каждое поле на схеме имеет символическое имя, которое служит для идентификации данного поля; данные имена задают смещения в буфере сообщения. Макрокоманда $OPCDEF опреде-

ляет имена полей, и все другие символические имена, которые можно указывать в качестве содержимого какого-нибудь поля.

На рис. 82 показан формат буфера сообщения для кода OPC¤_RQ_RQST.

Формат буфера сообщения для OPC¤_RQ_RQST

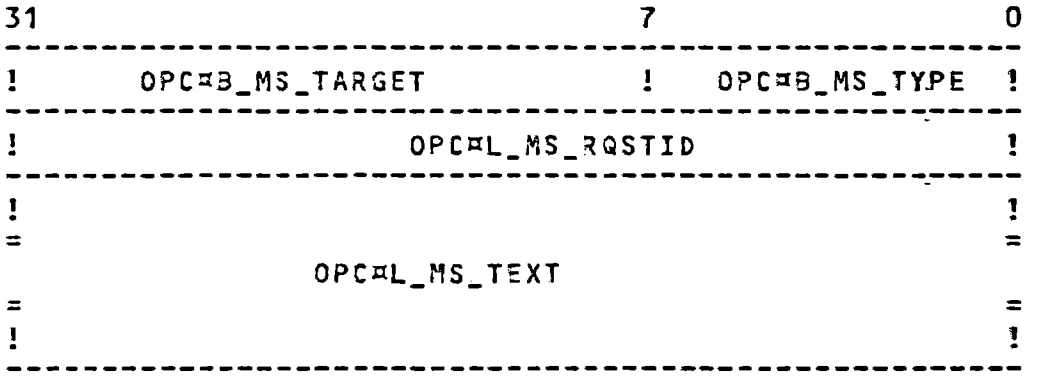


Рис. 82

Поля буфера сообщения:

OPC¤B_MS_TYPE - данное однобайтовое поле содержит код запроса OPC¤_RQ_RQST;

OPC¤B_MS_TARGET - данное трехбайтовое поле содержит 24-битовый вектор, который указывает, какие типы операторских терминалов должны получить данный запрос. Данный битовый вектор строится при помощи операции "логическое или" над указанными символическими именами требуемых типов. Следующий список дает символическое имя каждого типа операторского терминала:

OPC¤M_NM_CENTRL - центральный оператор;

OPC¤M_NM_DEVICE - информация о состоянии устройства;

OPC¤M_NM_DISKS - оператор дисковых устройств;

OPC¤M_NM_NTWORK - оператор сети;

OPC¤M_NM_TAPES - оператор ленточных устройств;

OPC¤M_NM_PRINT - оператор АЦПУ;

OPC¤M_NM_SECURITY - оператор системы защиты;

OPC¤M_NM_OPER1 - имена от OPC¤M_NM_OPER1 до OPC¤M_NM_OPER12 задают операторские функции, определенные администратором системы;

OPC¤M_NM_OPER12

OPC¤L_MS_RQSTID - это поле размером в длинное слово содержит обеспечиваемый пользователем код сообщения размером в длинное слово;

OPC¤L_MS_TEXT - это поле переменной длины содержит строку символов, задающую текст, который требуется послать на указанные операторские терминалы. Длина строки должна быть в диапазоне от 0 до 255 байтов.

Формат буфера сообщения для кода OPC¤_RQ_CANCEL показан на рис.83.

Формат буфера сообщения для OP¤_RQ_CANCEL

31	15	7	0
!	OPC¤B_MS_TARGET		!OPC¤B_MS_TYPE!
!	OPC¤L_MS_RQSTID		!

Рис. 83

Поля буфера сообщения:

OPC¤B_MS_TYPE - данное однобайтовое поле содержит код запроса OPC¤_RQ_CANCEL;

OPC¤B_MS_TARGET - данное трехбайтовое поле содержит 24-битовый вектор, который указывает, какие типы операторских терминалов должны получить запрос на отмену. Данный битовый вектор строится при помощи операции "логическое или" над указанными символическими именами требуемых типов. Следующий список дает символическое имя для каждого типа операторского терминала:

OPC¤M_NM_CENTRL - центральный оператор;

OPC¤M_NM_DEVICE - информация о состоянии устройства;

OPC¤M_NM_DISKS - оператор дисковых устройств;

OPC¤M_NM_NTWORK - оператор сети;

OPC¤M_NM_TAPES - оператор ленточных устройств;

OPC¤M_NM_PRINT - оператор АЦПУ;

OPC¤M_NM_SECURITY - оператор системы защиты;

OPC¤M_NM_OPER1 - имена от OPC¤M_NM_OPER1 до OPC¤M_NM_OPER12 задают операторские функции, определенные администратором системы;

OPC¤L_MS_RQSTID - данное поле размером в длинное слово содержит обеспечиваемый пользователем код сообщения размером в длинное слово.

Формат буфера сообщения для кода OPC¤_RQ_REPLY показан на рис.64.

Формат буфера сообщения для OPC$_RQ_REPLY

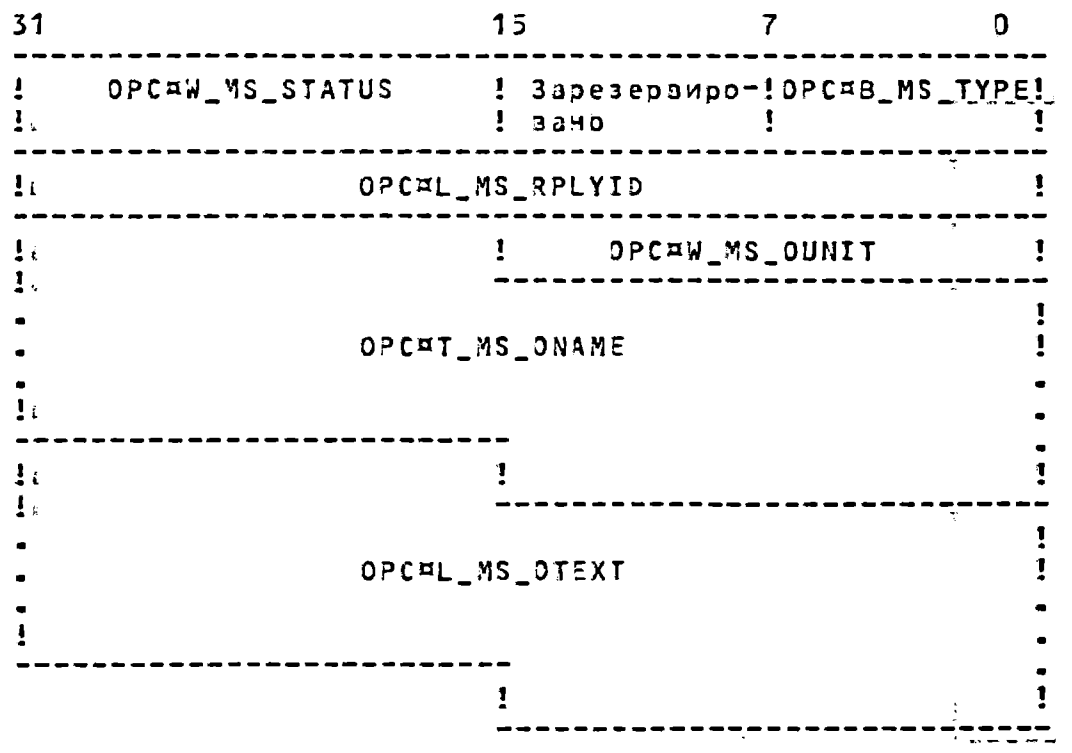


Рис. 84

Поля буфера сообщения:

OPC$B_MS_TYPE - данное однобайтовое поле содержит код запроса OPC$RQ_REPLY.

Зарезервировано - данное однобайтовое поле не используется;

OPC$W_MS_STATUS - данное двухбайтовое поле содержит младшее слово значения кода состояния размером в длинное слово, которое $SNDOPR возвращает в почтовый ящик, заданный аргументом CHAN. Чтобы проверить код завершения, необходимо извлечь младшее слово из длинного слова кода состояния и сравнить его с содержимым поля OPCW_MSSTATUS;

OPC$L_MS_PRLYID - данное 4-байтовое поле содержит обеспечиваемый пользователем код сообщения;

OPC¤W_MS_OUNIT - данное двухбайтовое поле содержит номер устройства терминала, на который необходимо послать ответ оператора. Чтобы получить данный номер устройства терминала, можно вызвать ¤GETDVI, задавая код элемента DVI¤_FULLDEVNAM. Возвращаемая информация будет содержать имя узла и имя устройства как сплошную строку. Поскольку номер устройства находится в конце данной строки, то необходимо сделать грамматический разбор строки. Это можно сделать одним способом - искать, начиная от конца, первый буквенный символ; цифры справа от буквенного символа составляют номер устройства. Из остатка строки затем строится строка символов со счетчиком, которая используется для следующего поля, OPC¤T_MS_ONAME;

OPC¤T_MS_ONAME - данное поле переменной длины содержит строку со счетчиком, задающую имя устройства терминала, который должен получить ответ оператора. Максимальная общая длина строки - 14 байтов;

OPC¤L_MS_OTEXT - данное поле переменной длины содержит строку символов, задающую написанный оператором текст, который необходимо послать на пользовательский терминал. Длина строки должна быть в диапазоне от 0 до 255 байтов. Данное поле не является обязательным.

Формат буфера сообщения для кода OPC¤RQ_TERME показан на рис.85.

Формат буфера сообщения для OPC_RQ_TERME

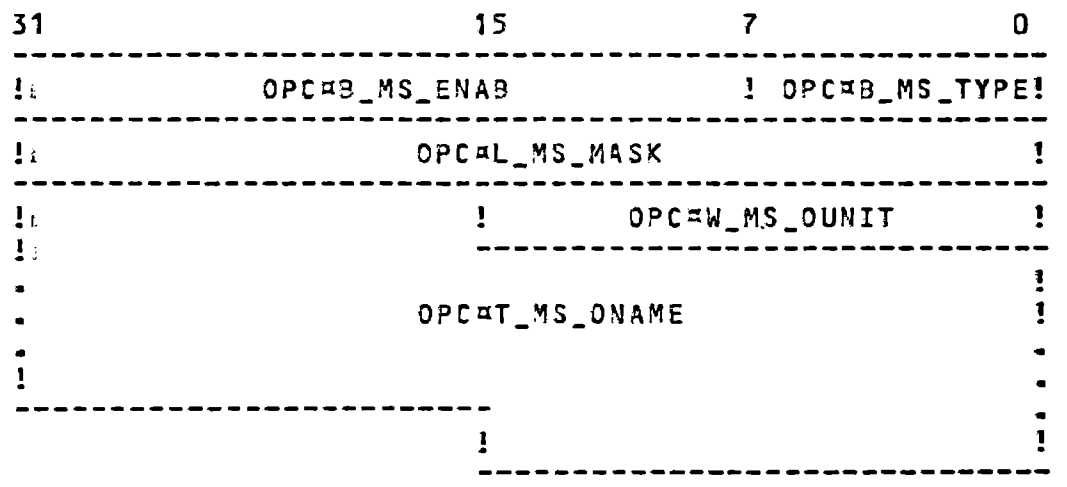


Рис. 85

Поля буфера сообщения:

OPC¤B_MS_TYPE - данное однобайтовое поле содержит код запроса OPC¤_RQ_TERME;

OPC¤B_MS_ENAB - данное трехбайтовое поле содержит обеспечиваемое пользователем значение. Значение 0 указывает на то, что заданный терминал необходимо запретить для указанного класса операторов. Любое ненулевое значение указывает на то, что заданный терминал необходимо разрешить для указанного класса операторов;

OPC¤B_MS_MASK - данное 4-байтовое поле содержит битовый вектор, который указывает, каким типам операторских терминалов необходимо разрешить или запретить использование заданного терминала. Битовый вектор строится при помощи операции "логическое или" над заданными символическими именами желаемых типов. Следующий список содержит символические имена каждого типа операторского терминала:

OPC¤M_NM_CENTRL - центральный оператор;

OPC¤M_NM_CESURITY - оператор системы защиты;

OPC¤M_NM_DEVICE - информация о состоянии устройства;

OPC¤M_NM_DISKS - оператор дисковых устройств;

OPC¤M_NM_NTWORK - оператор сети;

OPC¤M_NM_TAPES - оператор ленточных устройств;

OPC¤M_NM_PRINT - оператор ацпу;

OPC¤M_NM_OPER1 - имена от OPC¤M_NM_OPER1 до OPC¤M_NM_OPER12 задают операторские функции, определенные администратором системы;

OPC¤M_NM_OPER12

OPC¤W_MS_OUNIT - это двухбайтовое поле содержит номер устройства операторского терминала, который должен быть запрещен или разрешен для указанных типов операторских терминалов. Чтобы получить данный номер устройтсва терминала, необходимо вызвать ¤GETDVI, задавая код элемента DVI¤_FULLDEVNAM. Возвращаемая информация будет содержать имя узла и имя устройства как сплошную строку. Поскольку номер устройства находится в конце данной строки, то необходимо сделать грамматический разбор строки. Это можно сделать одним способом - искать, начиная от конца, первый буквенный символ; цифры справа от первого буквенного символа составляют номер устройства. Из остатка строки затем строится строка символов со счетчиком, которая используется для следующего поля, OPC¤T_MS_ONAME;

OPC¤T_MS_ONAME - это поле переменной длины содержит строку символов со счетчиком, задающую имя устройства опе-

раторского терминала, который необходимо запретить или разрешить для заданного типа операторского терминала. Максимальная общая длина строки - 16 байтов.

Формат буфера сообщения для кода OPC¤_RQ_STATUS показан на рис.86.

Формат буфера сообщения для OPC_RQ_STATUS

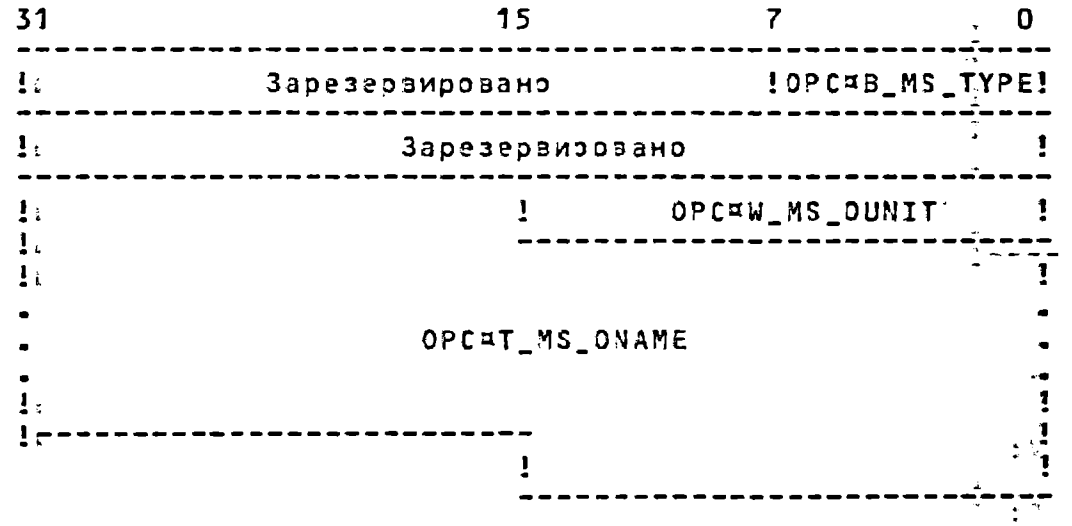


Рис. 86

Поля буфера сообщения:

OPC¤B_MS_TYPE - данное однобайтовое поле содержит код запроса OPC¤_RQ_STATUS;

Зарезервировано - данное трехбайтовое поле не используется;

Зарезервировано - данное четырехбайтовое поле не используется;

OPC¤W_MS_OUNIT - данное двухбайтовое поле содержит номер устройства операторского терминала, состояние которого требуется запросить. Чтобы получить данный номер, можно

вызвать $GETDVI, задавая код элемента DVI$_FULLDEVNAM. Возвращаемая информация будет содержать имя узла и имя устройства как сплошную строку. Поскольку номер устройства находится в конце данной строки, необходимо сделать грамматический разбор строки. Это можно сделать одним способом - искать, начиная от конца, первый буквенный символ; цифры справа от первого буквенного символа составляют номер устройства. Из остатка строки затем строится строка символов со счетчиком, которая используется для следующего поля, OPC$_MS_ONAME;

OPC$_MS_ONAME - данное поле переменной длины содержит строку символов со счетчиком, задающую имя устройства операторского терминала, состояние которого необходимо запросить. Максимальная общая длина строки - 14 байтов.

Формат буфера сообщения для кода OPC$_RQ_LOGI показан на рис.87.

Формат буфера сообщения для OPC_RQ_LOGI

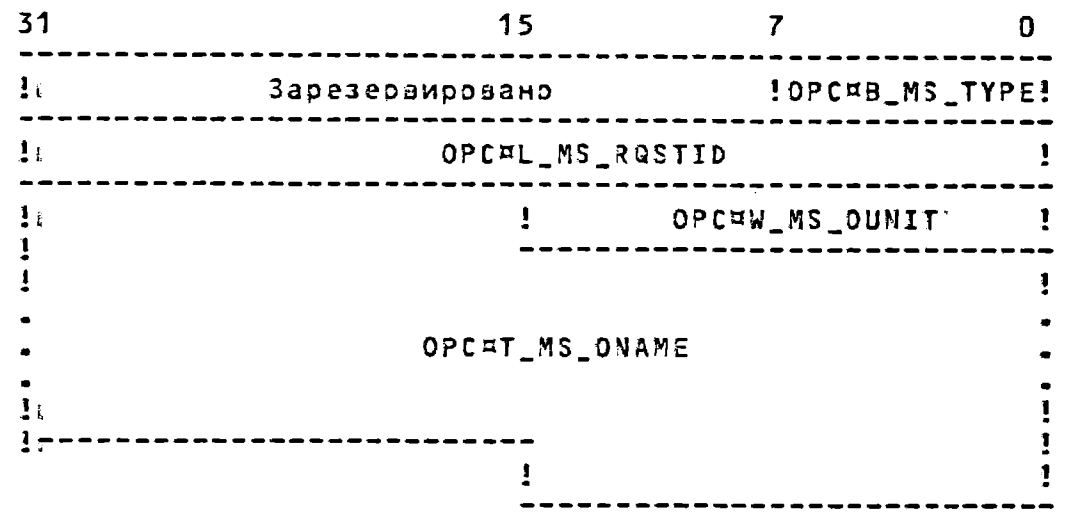


Рис. 87

Поля буфера сообщения:

OPC$B_MS_TYPE - данное однобайтовое поле содержит код запроса OPC$_RQ_LOGI;

Зарезервировано - данное трехбайтовое поле не используется;

OPC$L_MS_RQSTID - это поле размером в длинное слово содержит значение, обеспеченное пользователем. Значение 0 указывает, что текущий операторский файл регистрации необходимо закрыть, а новый файл регистрации - открыть. Значение 1 указывает, что текущий операторский файл регистрации необходимо закрыть, но новый файл регистрации открывать не требуется;

OPC$W_MS_OUNIT - данное двухбайтовое поле содержит номер устройства операторского терминала, который делает запрос на инициализацию. Чтобы получить данный номер устройства терминала, можно вызвать $GETDVI, задавая код эле-

мента DVI$_FULLDEVNAM. Возвращаемая информация будет содержать имя узла и имя устройства как сплошную строку. Поскольку номер устройства находится в конце данной строки, то необходимо сделать грамматический разбор строки. Это можно сделать одним способом - искать, начиная от конца, первый буквенный символ; цифры справа от первого буквенного символа составляют номер устройства. Из остатка строки затем строится строка символов со счетчиком, которая используется для следующего поля, OPC$_MS_ONAME;

OPC$_MS_ONAME - это поле переменной длины содержит строку символов со счетчиком, задающую имя устройства операторского терминала, который делает запрос на инициализацию. Максимальная общая длина строки - 14 байтов.

CHAN

Использование в МОС ВП: CHANNEL

Тип: слово (без знака)

Доступ: только чтение

Механизм: по значению

Канал, назначенный почтовому ящику, в который требуется послать ответ. Аргумент CHAN - это значение слова, содержащего данный номер канала. Если аргумент CHAN не указан или задан равным 0 (принимается по умолчанию), то ответ не посылается.

Аргумент CHAN необходимо задавать, если требуется ответ от оператора.

Описание:

В зависимости от конкретной операции использование

программы $SNDOPR может потребовать наличия у вызывающего процесса привилегии OPER для выполнения следующих функций:

1) разрешить использование терминала в качестве операторского;

2) ответить на пользовательский запрос или отменить его;

3) инициализировать файл регистрации операторских сообщений.

Кроме того, для выполнения функций, связанных с системой защиты, процесс должен иметь, кроме привилегии OPER, привилегию SECURITY.

Программа системного обслуживания "послать сообщение оператору" использует системную динамическую память.

В общем случае данная программа используется следующим образом:

Строится буфер сообщения и его окончательная длина помещается в первое слово дескриптора буфера;

Вызывается программа системного обслуживания $SNDOPR;

Проверяется код состояния, возвращаемый в регистре R0, для гарантии того, что вызов сделан успешно;

Выдается запрос на чтение почтового ящика, если почтовый ящик задан;

Когда чтение завершится, проверяется 2-байтовый код состояния в поле OPC$W_MS_STATUS для гарантии того, что операция была выполнена успешно.

Сообщения, отображаемые на операторские терминалы, имеют следующий формат:

00152-01 97 06

%%%%%%%%%% ОРСОМ ДД-МММ-ГГГГ ЧЧ:ММ:СС.СС информация конкретного сообщения

Возвращаемые значения кода состояния:

SS¤_NORMAL - успешное завершение;

SS¤_ACCVIO - вызывающая программа не может прочитать буфер сообщения или дескриптор буфера;

SS¤_BADPARAM - указанное сообщение имеет нулевую длину или длину более 986 байтов;

SS¤_DEVNOTMBX - указанный канал не назначен почтовому ящику;

SS¤_INSFMEM - для выполнения программы системного обслуживания не хватает системной динамической памяти;

SS¤_IVCHAN - указан недопустимый номер канала. Недопустимым является тот номер канала, котоый равен 0 или превышает количество доступных каналов;

SS¤_NOPRIV - у процесса нет привилегии, необходимой для того, чтобы ответить на запрос пользователя или отменить его; процесс не имеет доступа на чтение/запись к указанному почтовому ящику; или канал был назначен из более привилегированного режима доступа.

Коды состояния, возвращаемые в почтовый ящик:

OPC$_BLANKTAPE - успешное завершение; оператор ответил командой REPLY/BLANK_TAPE=N;

OPC$_INITAPE - успешное завершение; оператор ответил командой REPLY/INITIALIZE_TAPE=N;

OPC$_NOPERATOR - успешное завершение; нет операторского терминала, который может получить сообщение;

OPC$_RQSTCMPLTE - успешное завершение; оператор выполнил запрос;

OPC$_RQSTPEND - успешное завершение; оператор выполнит запрос, когда будет возможность;

OPC$_RQSTABORT - оператор не может удовлетворить запрос;

OPC$_RQSTCAN - вызывающий процесс отменил запрос.

13.109. SUSPND - приостановить процесс

Программа системного обслуживания SUSPND позволяет процессу приостановить себя или другой процесс. Приостановленный процесс не может получать AST или как-то иначе выполняться, до тех пор, пока другой процесс не возобновит или не удалит его.

Формат:

SYS$SUSPND [PIDADR],[PRCNAM]

Аргументы:

PIDADR

Использование в МОС ВП: PROCESS_ID

Тип: длинное слово (без знака)

Доступ: модификация

Механизм: по ссылке

Идентификатор процесса (PID), который необходимо приостановить. Аргумент PIDADR - это адрес длинного слова, содержащего PID.

Аргумент PIDADR обязателен, если необходимо приостановить процесс, номер группы UIC которого отличен от номера группы UIC вызывающего процесса.

PRCNAM

Использование в МОС ВП: PROCESS_NAME

Тип: строка текста

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки с фиксированной длиной

Имя процесса, который необходимо приостановить. Аргумент PRCNAM - это адрес дескриптора строки символов, указывающего на строку имени процесса длиной от 1 до 15 символов.

Имя процесса неявно квалифицируется его номером группы UIC. Поэтому аргумент PRCNAM можно использовать только для того, чтобы приостановить процесс, принадлежащий той же группе UIC, что и вызывающий процесс.

Для того, чтобы приостановить процессы в других груп-

пах, необходимо использовать аргумент PIDADR.

Если ни аргумент PIDADR, ни аргумент PRCNAM не указаны, то приостанавливается вызывающий процесс.

Описание:

В зависимости от конкретной операции использование программы ¤SUSPND может потребовать наличия у вызывающего процесса одной из перечисленных привилегий:

1) привилегия GROUP требуется для того, чтобы приостановить другой процесс в той же группе, если UIC приостанавливаемого процесса отличается от UIC вызываемого процесса;

2) привилегия WORLD необходима для того, чтобы приостановить любой процесс в системе.

Программа системного обслуживания ¤SUSPND требует системной динамической памяти.

Приостановленный процесс, если только он не имеет страниц, зафиксированных в памяти, можно удалить из балансного набора, чтобы дать возможность выполняться другим процессам.

Программа системного обслуживания "возобновить процесс" (¤RESUME) позволяет продолжить приостановленный процесс. Если для процесса, который не был приостановлен, выдан один или несколько запросов на возобновление, то следующий запрос на приостановку процесса сразу же завершается, то есть, процесс не приостанавливается.

Для выставленных запросов на возобновление счетчик не ведется.

Возвращаемые значения кода состояния:

SS¤_NORMAL - успешное завершение;

SS¤_ACCVIO - вызывающая программа не может прочитать строку имени процесса или дескриптор строки, либо не может записать идентификатор процесса;

SS¤_INSFMEM - недостаточно системной динамической памяти для выполнения программы системного обслуживания;

SS¤_IVLOGNAM - заданное имя процесса имеет нулевую длину или длину более 15 символов;

SS¤_NONEXPR - предупреждение; указанный процесс не существует, или была задана неверная идентификация процесса;

SS¤_NOPRIV - заданный процесс был создан не вызывающим процессом, а вызывающий процесс не имеет привилегии GROUP или WORLD.

13.110. ¤SYNCH - синхронизировать

Программа системного обслуживания ¤SYNCH проверяет состояние завершения программы системного обслуживания, которая завершается асинхронно.

Программа системного обслуживания, чье состояние завершения контролируется, должна вызываться с явно заданными аргументами EFN и IOSB, поскольку программа ¤SYNCH использует флаг события и блок состояния ввода-вывода контролируемой программы системного обслуживания.

В подразделе 2.5 подробно обсуждается завершение программ системного обслуживания.

Формат:

SYS$SYNCH [EFN],[IOSB]

Аргументы:

EFN

Использование в МОС ВП: EF_NUMBER

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Номер флага события, заданный в вызове программы системного обслуживания, чье состояние завершения должно проверяться программой $SYNCH. Аргумент EFN является длинным словом, содержащим данный номер.

IOSB

Использование в МОС ВП: IO_STATUS_BLOCK

Тип: квадрослово (без знака)

Доступ: только чтение

Механизм: по ссылке

Блок состояния ввода-вывода, заданный в вызове программы системного обслуживания, чье состояние должно проверяться программой $SYNCH. Аргумент IOSB является адресом данного квадрослова блока состояния ввода-вывода.

Описание:

Программа системного обслуживания выполняет действительный тест на завершение асинхронной программы системного обслуживания такой как, например, $GETJPI.

программа ¤SYNCH работает следующим образом:

После вызова программа ¤SYNCH ожидает (вызывает программу системного обслуживания ¤WAITFR) установки флага события;

После установки флага события программа ¤SYNCH проверяет содержимое блока состояния ввода-вывода. Если данное содержимое ненулевое, то, следовательно, асинхронная программа системного обслуживания действительно завершилась, и программа ¤SYNCH возвращает управление вызывающему процессу;

Если блок состояния ввода-вывода содержит нуль, то асинхронная программа системного обслуживания еще не завершилась, а флаг события был установлен завершением события, не связанного с завершением программы ¤GETJPI. В данном случае программа ¤SYNCH сбрасывает флаг события (вызывая программу ¤CLREF) и снова ожидает установки данного флага события (вызывая программу ¤WAITFR). Данный цикл повторяется до тех пор, пока содержимое блока состояния ввода-вывода не будет отличаться от нуля.

Программа системного обслуживания ¤SYNCH всегда устанавливает указанный флаг события, когда она возвращает управление вызывающему процессу. Это гарантирует, что другие сегменты программы смогут использовать этот же флаг события без возникновения конфликтных ситуаций. Предположим, что вызовы двух программ ¤GETJPI и ¤GETSYI задают один и тот же флаг события, а программа ¤SYNCH вызывается для проверки завершения из программы ¤GETJPI. Если данный флаг

события устанавливается программой ¤GETSYI, то программа ¤SYNCH сбрасывает данный флаг и ожидает, пока он будет установлен программой ¤GETJPI. Когда программа ¤GETJPI устанавливает флаг, программа ¤SYNCH возвращает управление вызывающему процессу и устанавливает данный флаг события. При таком способе работы установка флага программой ¤GETSYI не будет потеряна и другой вызов программы ¤SYNCH покажет завершение программы ¤GETSYI.

Программа системного обслуживания ¤SYNCH полезна, когда программа вызывает асинхронную программу системного обслуживания, но должна выполнить некоторую другую работу перед тем, как проверить завершение асинхронной программы. В данном случае программа должна вызывать программу ¤SYNCH в той точке, где она должна знать, что асинхронная программа завершилась и где требуется подождать действительного завершения данной программы.

Если программа вызывает асинхронную программу системного обслуживания (например, ¤QIO) и действительно ожидает (с помощью вызова программы ¤WAITFR) ее завершения, не выполняя при этом никакой другой работы, то данную программу можно было бы улучшить, если использовать синхронную форму данной программы системного обслуживания (например, ¤QIOW). Синхронные программы системного обслуживания, такие как ¤QIOW проверяют действительное состояние завершения тем же способом, как это делает программа ¤SYNCH.

Возвращаемые значения кодов состояния:

SS$_NORMAL - программа системного обслуживания завершилась. Асинхронная программа системного обслуживания завершилась и блок состояния ввода-вывода содержит значение кода состояния, описывающее состояние завершения асинхронной программы системного обслуживания.

13.111. $TRNLNM - транслировать логическое имя

Программа системного обслуживания $TRNLNM возвращает информацию о логическом имени.

Формат:

SYS$TRNLNM [ATTR],TABNAM,LOGNAM,[ACMODE],[ITMLST]

Аргументы:

ATTR

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Атрибуты контролирующие поиск логического имени. Аргумент ATTR является адресом длинного слова, содержащего битовую маску, которая задает данные атрибуты.

Каждый бит в длинном слове соответствует атрибуту и имеет символическое имя. Чтобы задать атрибут, необходимо задать символическое имя или установить соответствующий ему бит. Все неопределенные биты в длинном слове должны быть

сброшены на 0.

Если данный аргумент не задан или задан равным 0 (нет установленных битов), то не используются никакие атрибуты.

Макрокоманда $LNMDEF определяет следующее имя атрибута:

LNM$M_CASE_BLIND - если данный бит установлен, программа $TRNLNM не делает различия между большими и строчными буквами в транслируемом логическом имени.

TABNAM

Использование в МОС ВП: LOGICAL_NAME

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Имя таблицы или имя списка таблиц, в котором необходимо выполнить поиск логического имени. Аргумент TABNAM является адресом дескриптора, указывающего на данное имя. Это обязательный аргумент.

Если имя таблицы не является именем таблицы логических имен, то полагается, что данное логическое имя и транслируется итеративно до тех пор, пока либо не будет найдено имя таблицы логических имен, либо не будет выполнено допустимое системой число трансляций. Если имя таблицы транслируется в список таблиц логических имен, то таблицы просматриваются в указанном порядке.

LOGNAM

Использование в МОС ВП: LOGICAL_NAME

Тип: текстовая строка

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки фиксированной длины

Логическое имя, о котором должна вернуться информация. Аргумент LOGNAM является адресом дескриптора, указывающего на строку логического имени. Это обязательный аргумент.

ACMODE

Использование в МОС ВП: ACCESS_MODE

Тип: байт (без знака)

Доступ: только чтение

Механизм: по ссылке

Режим доступа, который должен использоваться при трансляции. Аргумент ACMODE является адресом байта, задающего режим доступа. Макрокоманда $PSLDEF определяет символические имена для четырех режимов доступа.

Если аргумент ACMODE задан, то программа $TRNLNM игнорирует все имена (как логические имена, так и имена таблиц) в режимах доступа менее привилегированных, чем указанный режим доступа. Указанный режим доступа не сравнивается с режимом доступа вызывающего процесса.

Если аргумент ACMODE не задан, то программа $TRNLNM выполняет трансляцию, не обращая внимания на режим доступа, однако процесс трансляции продолжается от самого внешнего режима доступа к самому внутреннему режиму доступа. Таким

образом, если в одной таблице имеются два логических имени с одинаковым именем, но разными режимами доступа, то программа $TRNLNM будет транслировать имя с самым внешним режимом доступа.

ITMLST

Использование в МОС ВП: ITEM_LIST_3

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по ссылке

Список элементов, описывающий, какую информацию должна возвращать программа $TRNLNM. Аргумент ITMLST является адресом списка дескрипторов элементов, каждый из которых задает или контролирует элемент информации, которая возвращается программой. Список дескрипторов элементов завершается длинным словом, содержащим 0.

Рис. 88 описывает формат дескриптора элемента.

Формат дескриптора элемента

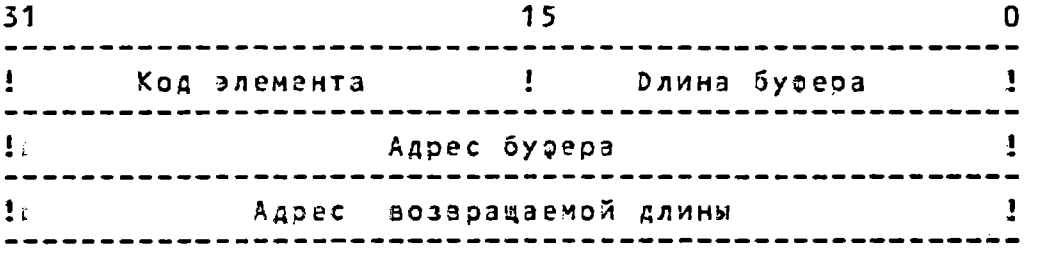


Рис. 88

Поля элементного дескриптора программы $TRNLNM:

длина буфера -

Слово, задающее число байтов в буфере, на который указывает поле "адрес буфера";

код элемента -

Слово, содержащее символический код, описывающий сущность информации в буфере, или возвращаемой в буфер, на который указывает поле "адрес буфера";

адрес буфера -

Длинное слово, содержащее адрес буфера, который задает или принимает информацию;

адрес возвращаемой длины -

Длинное слово, содержащее адрес слова, которое задает действительную длину в байтах информации, возвращаемой программой $TRNLNM в буфер, на который указывает поле "адрес буфера".

Коды элементов программы $TRNLNM:

LNM$_ACMODE -

Если задан данный код элемента, то программа возвращает режим доступа, связанный с данным логическим именем во время его создания. Поле "адрес буфера" дескриптора элемента является адресом байта, в который программа $TRNLNM записывает режим доступа;

LNM$_ATTRIBUTES -

Если задан данный код элемента, то программа возвращает атрибуты логического имени и эквивалентное имя, связанное с текущим значением кода LNM$_INDEX.

Поле "адрес буфера" дескриптора элемента указывает на длинное слово, содержащее битовую маску, в который каждый бит соответствует атрибуту. Программа системного обслуживания ¤TRNLNM устанавливает каждый бит для каждого атрибута, которым владеет либо логическое имя, либо эквивалентное имя.

Символические имена для данных атрибутов определяются макрокомандой ¤LNMDEF:

LNM¤M_CONCEALED -

Если программа устанавливает данный бит, то эквивалентное имя при текущем значении индекса для логического имени является скрытым логическим именем, как это интерпретируется системой СУД-32;

LNM¤M_CONFINE -

Если программа устанавливает данный бит, то логическое имя не копируется из процесса ни в один из порожденных им подпроцессов. Команда SPAWN языка DCL создает подпроцессы;

LNM¤M_CRELOG -

Если программа устанавливает данный бит, то логическое имя создано с использованием программы системного обслуживания ¤CRELOG;

LNM¤M_EXISTS -

Если программа устанавливает данный бит, то существует эквивалентное имя с заданным индексом;

LNM¤M_NO_ALIAS -

Если программа устанавливает данный бит, то имя логи-

ческого имени не может быть дано другому логическому имени, определенному в той же таблице для внешнего режима доступа;

LNM$M_TABLE -

Если программа устанавливает данный бит, то логическое имя является именем таблицы логических имен;

LNM$M_TERMINAL -

Если программа устанавливает данный бит, то эквивалентное имя для логического имени не может быть подвергнуто дальнейшей (рекурсивной) трансляции;

LNM$_CHAIN -

Если задан данный код элемента, то программа обрабатывает другой список элементов, который непосредственно следует за текущим списком. Поле "адрес буфера" дескриптора элемента указывает на следующий список элементов;

LNM$_INDEX -

Если задан данный код элемента, то программа ищет эквивалентное имя, имеющее указанное значение индекса. Поле "адрес буфера" дескриптора элемента указывает на длинное слово, содержащее заданное пользователем целое число в диапазоне от 0 до 127. Если данный код не задан. То подразумевается значение кода LNM$_INDEX равное 0 и программа $TRNLNM возвращает информацию об эквивалентном имени с индексом 0. Поскольку логическое имя может иметь более одного эквивалентного имени и каждое эквивалентное имя определяется значением индек-

са, то следует задавать код LNM¤_INDEX первым в списке элементов, перед тем как указывать коды LNM¤_STRING, LNM¤_LENGTH или LNM¤_ATTRIBUTES, поскольку данные коды возвращают информацию об эквивалентном имени, которое идентифицируется текущим значением индекса LNM¤_INDEX;

LNM¤_LENGTH -

Если задан данный код элемента, то программа возвращает действительную длину строки эквивалентного имени, соответствующего текущему значению эквивалентного имени. Поле "адрес буфера" дескриптора элемента является адресом длинного слова, в которое программа ¤TRNLNM записывает данную длину. Если для текущего значения кода LNM¤_INDEX не существует эквивалентного имени, то программа возвращает нуль в поле, на которое указывает поле "адрес возвращаемой длины" дескриптора элемента;

LNM¤_MAX_INDEX -

Каждое эквивалентное имя для логического имени имеет связанный с ним индекс. Если задан данный код элемента, то программа возвращает значение равное максимальному значению индекса эквивалентного имени. Поле "адрес буфера" дескриптора элемента является адресом длинного слова, в которое программа ¤TRNLNM записывает данное значение. Если нет никаких эквивалентных имен (и, следовательно, нет значений индексов), то программа ¤TRNLNM возвращает значение минус 1;

LNM¤_TABLE -

Если задан данный код элемента, то программа возвра-

дает имя таблицы, содержащей логическое имя, которое должно транслироваться. Поле "адрес буфера" дескриптора элемента указывает на буфер, в который программа $TRNLNM возвращает данное имя. Поле "адрес возвращаемой длины" дескриптора элемента задает адрес слова, в которое программа $TRNLNM пишет размер имени таблицы. Максимальная длина имен таблицы равна 31 символу.

LNM$_STRING -

Если задан данный код элемента, то программа возвращает строку эквивалентного имени, соответствующего текущему значению LNM$_INDEX. Поле "адрес буфера" дескриптора элемента указывает на буфер, содержащий данную строку. Поле "адрес возвращаемой длины" дескриптора элемента содержит адрес слова, которое содержит длину в байтах данной строки. Максимальная длина строки эквивалентного имени составляет 255 символов. Если для текущего значения кода LNM$_INDEX не существует эквивалентного имени, то программа $TRNLNM возвращает нулевое значение в поле "адрес возвращаемой длины" дескриптора элемента.

Описание:

Для транслирования логического имени, расположенного в разделяемой таблице логических имен требуется доступ с чтением в данной таблице.

Возвращаемые значения кода состояния:

SS¤_ACCVIO - программа системного обслуживания не может получить доступ к ячейкам памяти, заданным одним или более аргументами;

SS¤_BADPARAM - один или более аргументов имеют неверное значение, или не задано имя таблицы логических имен или логическое имя;

SS¤_BUFFEROVF - успешное завершение. Поле "длина буфера" дескриптора элемента задает недостаточное значение, чтобы вместить требуемые данные;

SS¤_IVLOGNAM - аргумент TABNAM или аргумент LOGNAM задает строку, длина которой лежит вне требуемого диапазона от 1 до 255 символов;

SS¤_IVLOGTAB - аргумент TABNAM не указывает на таблицу логических имен;

SS¤_NOLOGNAM - логическое имя не найдено в указанной таблице (или таблицах) логических имен;

SS¤_NOPRIV - у вызывающего процесса нет необходимой привилегии для доступа к указанному имени;

SS¤_NORMAL - успешное завершение. Найдено эквивалентное имя для логического имени;

SS¤_TOOMANYLNAM - трансляция логического имени для имени таблицы превысила допустимую глубину (10 трансляций).

13.112. ¤ULKPAG - освободить страницы в памяти

Программа системного обслуживания ¤ULKPAG освобождает страницы, которые были раньше зафиксированы в памяти при помощи программы системного обслуживания "зафиксировать страницы в памяти" (¤LCKPAG). Страницы автоматически освобождаются и удаляются при выходе образа.

Формат:

SYS¤ULKPAG INADR, [RETADR], [ACMODE]

Аргументы:

INADR

использование в MOC ВП: ADDRESS_RANGE

тип: длинное слово (без знака)

доступ: только чтение

механизм: по ссылке

Начальный и конечный виртуальные адреса освобождаемых страниц. Аргумент INADR - это адрес массива из двух длинных слов, содержащих по порядку начальный и конечный виртуальные адреса процесса. В каждом виртуальном адресе используется только та часть, которая представляет номер виртуальной страницы; младшие 9 битов игнорируются. Если начальный виртуальный адрес равен конечному, то освобождается одна страница.

Если освобождается несколько страниц, и необходимо конкретно определить, какие страницы были прежде свободны, то следует освобождать страницы по одной, то есть, одну страницу за один вызов $ULKPAG. Код состояния, возвращаемый программой $ULKPAG, покажет была ли данная страница прежде освобожденной.

RETADR

использование в МОС ВП: ADDRESS_RANGE

тип: длинное слово (без знака)

доступ: только запись

механизм: по ссылке

Начальный и конечный виртуальные адреса страниц, фактически освобожденных программой $ULKPAG. Аргумент RETADR - это адрес массива из двух длинных слов, содержащих по порядку начальный и конечный виртуальные адреса процесса.

Если во время освобождения нескольких страниц возникла ошибка, то RETADR показывает те страницы, которые были успешно освобождены до возникновения ошибки. Если не было успешно освобожденных страниц, то оба длинных слова в массиве RETADR будут содержать значение минус 1.

ACMODE

использование в МОС ВП: ACCESS_MODE

тип: длинное слово (без знака)

доступ: только чтение

механизм: по значению

Режим доступа, для которого делается данный запрос. Аргумент ACMODE - это длинное слово, содержащее режим дос-

тупа. Макрокоманда $PSLDEF определяет обозначения для четырех режимов доступа.

Наиболее привилегированный режим, который может использоваться, - это режим доступа вызывающей программы.

Для освобождения какой-либо указанной страницы результирующий режим доступа должен иметь равную или высшую привилегию по сравнению с режимом доступа владельца данной страницы.

Описание:

Для того, чтобы вызвать программу системного обслуживания $ULKPAG, процесс должен иметь привилегию PSWAPM.

Возвращаемые значения кода состояния:

SS$_WASCLR - успешное завершение. По крайней мере одна из указанных страниц была ранее освобождена;

SS$_WASSET - успешное завершение. Все заданные страницы были раньше зафиксированы;

SS$_ACCVIO - вызывающая программа не может прочитать входной массив или не может записать выходной массив; либо какая-то страница в указанном диапазоне недоступна или не существует.

13.113. $ULWSET - освободить страницы в рабочем наборе

Программа системного обслуживания $ULWSET освобождает страницы, которые были раньше зафиксированы в рабочем набо-

ре при помощи программы системного обслуживания "зафиксировать страницы в рабочем наборе" (¤LKWSET). Освобожденные страницы становятся кандидатами на замещение внутри рабочего набора процесса.

Формат:

SYS¤ULWSET INADR, [RETADR], [ACMODE]

Аргументы:

INADR

использование в МОС ВП: ADDRESS_RANGE

тип: длинное слово (без знака)

доступ: только чтение

механизм: по ссылке

Начальный и конечный виртуальные адреса освобождаемых страниц. Аргумент INADR - это адрес массива из двух длинных слов, содержащих по порядку начальный и конечный виртуальные адреса процесса. В каждом виртуальном адресе используется только та часть, которая представляет номер виртуальной страницы; младшие 9 битов игнорируются. Если начальный виртуальный адрес равен конечному, то освобождается одна страница.

Если освобождается несколько страниц, и необходимо конкретно определить, какие страницы были прежде свободны, то следует освобождать страницы по одной, то есть, одну страницу за один вызов ¤ULWSET. Код состояния, возвращаемый программой ¤ULWSET, покажет, была ли данная страница прежде освобожденной.

RETADR

использование в МОС ВП: ADDRESS_RANGE

тип: длинное слово (без знака)

доступ: только запись

механизм: по ссылке

Начальный и конечный виртуальные адреса страниц, фактически освобожденных программой $ULWSET. Аргумент RETADR - это адрес массива из двух длинных слов, содержащих по порядку начальный и конечный виртуальные адреса процесса.

Если во время освобождения нескольких страниц возникла ошибка, то RETADR показывает те страницы, которые были успешно освобождены до возникновения ошибки. Если не было успешно освобожденных страниц, то оба длинных слова аргумента RETADR будут содержать значение "минус 1".

ACMODE

использование в МОС ВП: ACCESS_MODE

тип: длинное слово (без знака)

доступ: только чтение

механизм: по значению

Режим доступа, для которого делается данный запрос. Аргумент ACMODE - это длинное слово, содержащее режим доступа. Макрокоманда $PSLDEF определяет обозначения для четырех режимов доступа.

Наиболее привилегированный режим, который может использоваться, - это режим доступа вызывающей программы. Для освобождения какой-либо указанной страницы результирующий режим доступа должен иметь равную или высшую привилегию

по сравнению с режимом доступа владельца данной страницы.

Возвращаемые значения кода состояния:

SS$_WASCLR - успешное завершение. По крайней мере одна из указанных страниц была раньше освобождена;

SS$_WASSET - успешное завершение. Все указанные страницы были раньше захвачены в рабочем наборе;

SS$_ACCVIO - вызывающая программа не может прочитать аргумент INADR или не может записать аргумент RETADR; либо страница в заданном диапазоне недоступна или не существует;

SS$_NOPRIV - страница в указанном диапазоне принадлежит системному адресному пространству.

13.114. $UNWIND - развернуть стек вызовов

Программа системного обслуживания $UNWIND разворачивает стек вызовов программ; то есть, она удаляет заданное количество кадров вызова из стека. По требованию она может после развертывания стека вернуть управление по адресу нового счетчика инструкций (PC). Предполагается, что $UNWIND должна вызываться из программы обработки кода состояния.

Формат:

SYS$UNWIND [DEPADR], [NEWPC]

Аргументы:

DEPADR

использование в МОС ВП: LONGWORD_UNSIGNED

тип: длинное слово (без знака)

доступ: только чтение

механизм: по ссылке

Глубина, на которую необходимо развернуть стек вызовов программ. Аргумент DEPADR - это адрес значения длинного слова. Значение 0 задает кадр вызова программы, которая выполнялась в тот момент, когда возникло данное состояние (то есть, кадры вызовов не разворачиваются); значение 1 задает кадр программы, вызывающей тот кадр; значение 2 задает кадр программы, вызывающей программу, которая вызывает тот кадр, и т.д.

Если аргумент DEPADR задан как 0, то развертывания не происходит, а $UNWIND возвращает в регистре R0 значение кода успешного завершения.

Если аргумент DEPADR не задан, то $UNWIND развертывает стек до кадра вызова программы, вызвавшей ту программу, которая организовала обработчик кода состояния, вызвавший программу системного обслуживания $UNWIND. Данный метод развертывания стека вызовов процедур является обычным и принятым по умолчанию.

NEWPC

использование в МОС ВП: ADDRESS

тип: длинное слово (без знака)

доступ: только чтение

механизм: по ссылке

Новое значение счетчика инструкций (PC); данное значение замещает текущее значение PC в кадре вызова той программы, которая получит управление по завершении операции развертывания. Аргумент NEWPC - это значение длинного слова, содержащего адрес, с которого необходимо возобновить выполнение.

Выполнение возобновляется с данного адреса после того, как завершится операция развертывания.

Если аргумент NEWPC не задан, то выполнение возобновляется с адреса, задаваемого счетчиком инструкций в кадре вызова той программы, которая получит управление по завершении операции развертывания.

Описание:

Фактическое развертывание выполняется не сразу. Сначала адреса возврата в стеке вызовов $UNWIND проверяет каждый кадр в стеке вызовов на наличие об'явленного обработчика кода состояния. Если обработчик об'явлен, $UNWIND вызывает данный обработчик, задавая в качестве аргумента "имя состояния" в сигнальном массиве значения кода состояния SS$_UNWIND (указывающее на то, что осуществляется развертывание стека вызовов). Если вызываемый обработчик кода состояния получает данное значение кода состояния, он может

выполнить какие-либо необходимые операции очистки, связанные со спецификой программы. После того, как обработчик кода состояния возвратит управление, данный кадр вызова удаляется из стека.

Возвращаемые значения кода состояния:

SS$_NORMAL - успешное завершение;

SS$_ACCVIO - стек вызовов недоступен для вызывающей программы. Данная ситуация обнаруживается при просмотре стеков вызовов с целью модификации адресов возврата;

SS$_INSFRAME - не хватает кадров вызова для того, чтобы осуществить развертывание на заданную глубину;

SS$_NOSIGNAL - предупреждение. В текущий момент нет активных сигналов о состоянии исключительной ситуации;

SS$_UNWINDING - предупреждение. Операция развертывания уже находится в развитии.

13.115. $UPDSEC - обновить файл секции на диске

Программа системного обслуживания $UPDSEC записывает все модифицированные страницы в активной личной или глобальной секции обратно в файл секции на диске. В очередь ставится один или несколько запросов на операцию ввода-вывода, в зависимости от количества модифицированных страниц.

Формат:

SYS$UPDSEC INADR,[RETADR],[ACMODE],[UPDFLG]
,[EFN],[IOSB],[ASTADR],[ASTPRM]

Аргументы:

INADR

использование в МОС ВП: ADDRESS_RANGE

тип: длинное слово (без знака)

доступ: только чтение

механизм: по ссылке

Начальный и конечный виртуальные адреса страниц, которые необходимо записать в файл секции, если они были модифицированы. Аргумент INADR - это адрес массива из двух длинных слов, содержащих по порядку начальный и конечный виртуальные адреса процесса. В каждом виртуальном адресе используется только та часть, которая представляет номер виртуальной страницы; младшие 9 битов игнорируются.

Программа системного обслуживания $UPDSEC просматривает страницы, начиная с адреса, содержащегося в первом длинном слове, заданном аргументом INADR, и заканчивая адресом, содержащимся во втором длинном слове. Внутри данного диапазона $UPDSEC локализует страницы с доступом на чтение/запись, которые были модифицированы, и записывает их (по возможности, друг за другом) в файл секции на диске. Немодифицированные страницы также перезаписываются на диск, если они разделяют один и тот же кластер с модифицированными страницами.

Если начальный виртуальный адрес равен конечному, то в

файл секции записывается одна страница, если она была модифицирована.

Адрес, заданный вторым длинным словом, может быть меньше адреса, заданного первым длинным словом.

RETADR

использование в МОС ВП: ADDRESS_RANGE

тип: длинное слово (без знака)

доступ: только запись

механизм: по ссылке

Адрес первой и последней страниц, которые были фактически поставлены в очередь на запись обратно в файл секции на диске в первом запросе $QIO. Аргумент RETADR - это адрес массива из двух длинных слов, содержащих по порядку адрес первой страницы и адрес последней страницы.

Если $UPDSEC возвращает в регистре R0 значение кода состояния ошибки, то каждое длинное слово, заданное аргументом RETADR, будет содержать значение "минус 1". В данном случае флаг события не устанавливается, AST не доставляется и блок состояния ввода-вывода не записывается.

ACMODE

использование в МОС ВП: ACCESS_MODE

тип: длинное слово (без знака)

доступ: только чтение

механизм: по значению

Режим доступа, для которого выполняется данная программа системного обслуживания. Аргумент ACMODE - это длинное слово, содержащее режим доступа. Макрокоманда $PSLDEF

определяет обозначения для четырех режимов доступа.

Наиболее привилегированный режим доступа, который может использоваться, - это режим доступа вызывающего процесса. Страницу нельзя записать на диск, если режим доступа, используемый программой $UPDSEC, не будет иметь равную или высшую привилегию по сравнению с режимом доступа владельца той страницы, которую необходимо записать.

UPDFLG

использование в МОС ВП: LONGWORD_UNSIDNED

тип: длинное слово (без знака)

доступ: только чтение

механизм: по значению

Указатель обновления для глобальных секций с доступом на чтение/запись. Аргумент UPDFLG - это значение размером в длинное слово. Значение 0 (принимаемое по умолчанию) указывает, что все страницы с доступом на чтение/запись в данной глобальной секции, необходимо записать в файл секции на диске, независимо от того, были они модифицированы или нет. Значение 1 указывает, что вызывающий процесс является единственным процессом, который фактически пишет в глобальную секцию и в файл секции на диск необходимо записать только те страницы, которые были фактически модифицированы вызывающим процессом.

EFN

использование в МОС ВП: EFN_NUMBER

тип: длинное слово (без знака)

доступ: только чтение

механизм: по значению

Флаг события, который необходимо установить, когда файл секции на диске будет фактически обновлен. Аргумен EFN - это длинное слово, задающее номер флага события.

Если EFN не задан, то используется флаг события 0.

Когда $UPDSEC вызывается, то заданный флаг события или флаг события 0 очищается; когда операция обновления завершается, то флаг события устанавливается в 1.

IOSB

использование в МОС ВП: IO_STATUS_BLOCK

тип: квадрослово (без знака)

доступ: только запись

механизм: по ссылке

Блок состояния ввода-вывода, который должен получить окончательный код завершения операции обновления. Аргумент IOSB - это адрес квадрослова, содержащего блок состояния ввода-вывода.

При вызове $UPDSEC блок состояния ввода-вывода очищается. Как только операция обновления завершается, то есть, завершаются все операции ввода-вывода на диск, в блок состояния ввода-вывода записывается следующее:

1) первое слово содержит код состояния, возвращаемый программой $QIO, который показывает окончательное состояние завершения;

2) первый бит второго слова будет установлен только в том случае, если во время операции ввода-вывода возникла ошибка, и это была аппаратная ошибка при записи;

3) второе слово содержит виртуальный адрес первой незаписанной страницы.

Хотя данный аргумент не является обязательным, рекомендуется указывать его по следующим причинам:

1) если для сигнализации о завершении программы системного обслуживания используется флаг события, то можно проверить код состояния в блоке состояния ввода-вывода, чтобы убедиться, что флаг события не был установлен событием, отличным от завершения программы системного обслуживания;

2) если для синхронизации завершения программы системного обслуживания используется программа системного обслуживания ¤SYNCH, то блок состояния ввода-вывода является обязательным аргументом для ¤SYNCH;

3) код состояния, возвращаемый в R0, и код состояния, возвращаемый в блок состояния ввода-вывода, обеспечивает информацию о разных аспектах вызова программы системного обслуживания ¤UPDSEC. Код состояния, возвращаемый в R0, дает информацию о том, насколько успешно осуществился сам вызов программы системного обслуживания; код состояния, возвращаемый в блок состояния ввода-вывода, дает информацию о том, насколько успешной была работа программы системного обслуживания. Следовательно, чтобы точно оценить, насколько успешным был вызов ¤UPDSEC, необходимо проверить коды состояния, которые возвращаются и в R0 , и в блок состояния ввода-вывода.

ASTADR

использование в МОС ВП: AST_PROCEDURE

тип: маска входа процедуры

доступ: вызов без развертывания стека

механизм: по ссылке

Подпрограмма обработки AST, которую необходимо выполнить, когда файл секции будет обновлен. Аргумент ASTADR - это адрес маски входа данной подпрограммы.

Если задан аргумент ASTADR, то данная подпрограмма выполняется в том же режиме доступа, из которого было запрошено обновление файла секции.

ASTPRM

использование в МОС ВП: USER_ARG

тип: длинное слово (без знака)

доступ: только чтение

механизм: по значению

Параметр AST, который необходимо передать подпрограмме AST. Аргумент ASTPRM - это параметр размером в длинное слово.

Описание.

Программа системного обслуживания $UPDSEC использует квоту предела прямого ввода-вывода (DIRIO) вызывающего процесса при постановке в очередь запроса на операцию ввода-вывода и, если задан аргумент ASTADR, использует квоту предел AST (ASTLM) вызывающего процесса.

Для корректного использования данной программы системного обслуживания требуется, чтобы вызывающий процесс синх-

ронизировал завершение запроса на обновление. Для этого сначала необходимо проверить код состояния, возвращаемый программой $UPDSEC в регистре R0. Если возвращено значение SS$_NOTMODIFIED, вызывающий процесс может продолжать работу. Если возвращено значение SS$_NORMAL, то вызывающий процесс должен подождать завершения ввода-вывода и потом проверить окончательное состояние завершения в блоке состояния ввода-вывода. Для определения того, что операция ввода-вывода фактически завершилась, можно использовать программу системного обслуживания "синхронизировать" ($SYNCH).

Возвращаемые значения кода состояния:

SS$_NORMAL - успешное завершение. В очередь был поставлен один или несколько запросов на операцию ввода-вывода;

SS$_NOTMODIFIED - успешное завершение. В диапазоне входных адресов не оказалось модифицированных страниц. Запросы на операции ввода-вывода в очередь не ставились;

SS$_ACCVIO - вызывающая программа не может прочитать массив входных адресов или не может записать массив выходных адресов;

SS$_EXQUOTA - процесс превысил свою квоту предела AST;

SS$_ILLEFC - указан недопустимый номер флага события;

SS$_IVSECFLG - задан недопустимый флаг;

SS$_NOTCREATOR - секция размещается в памяти, разделяемой несколькими процессорами, и она была создана процессом, работающим на другом процессоре;

SS$_NOPRIV - страница в заданном диапазоне находится в системном адресном пространстве;

SS$_PAGOWNVIO - страница в заданном диапазоне принадлежит более привилегированному режиму доступа, чем режим доступа вызывающего процесса;

SS$_SHMNOTCNCT секция определена как секция, размещенная в разделяемой памяти, однако такая разделяемая память неизвестна системе;

SS$_UNASCEFC - процесс не связан с кластером, содержащим указанный флаг события.

13.116. $UPDSECW - обновить файл секции на диске и ждать

Программа системного обслуживания $UPDSECW записывает все модифицированные страницы в активной личной или глобальной секции обратно в файл секции на диске. В очередь ставится один или несколько запросов на операцию ввода-вывода, в зависимости от количества модифицированных страниц.

Программа системного обслуживания ¤UPDSECW завершается синхронно; то есть, она возвращает управление вызывающему процессу после записи всех обновленных страниц.

Для асинхронного завершения необходимо использовать программу системного обслуживания "обновить файл секции на диске". (¤UPDSEC); ¤UPDSEC возвращает управление вызывающему процессу после постановки в очередь запроса на обновление, не ожидая, когда страницы будут обновлены.

Во всех других отношениях программа ¤UPDSECW идентична ¤UPDSEC. Вся остальная информация о программе системного обслуживания ¤UPDSECW содержится в описании ¤UPDSEC (см. подраздел 2.5).

Формат:

```
SYS¤UPDSECW     INADR[,RETADR][,ACMODE][,UPDFLG]
                [,EFN][,IOSB][,ASTADR][,ASTPRM]
```

13.117. ¤WAITFR - ждать один флаг события

Программа системного обслуживания ¤WAITFR проверяет указанный флаг события и сразу же возвращает управление, если флаг установлен. В противном случае процесс переводится в состояние ожидания до тех пор, пока данный флаг не будет установлен.

Формат:

```
SYS¤WAITFR      EFN
```

Аргументы:

EFN

использование в MOC ВП: EF-NUMBER

тип: длинное слово (без знака)

доступ: только чтение

механизм: по значению

Номер флага события, установку которого контролирует программа системного обслуживания. Аргумент EFN является длинным словом, содержащим данный номер.

Описание:

Состояние ожидания, вызванное данной программой системного обслуживания, может быть прервано асинхронным системным прерыванием (AST), если режим доступа с которым выполняется AST равен или является более привилегированным, чем режим доступа процесса, вызвавшего программу $WAITFR и процессу разрешены прерывания в данном режиме доступа.

Если состояние ожидания прерывается прерыванием AST, то после того как подпрограмма обработки AST завершает выполнение, операционная система MOC ВП повторяет запрос программы $WAITFR для процесса. В данной точке, если флаг события установлен, то процесс возобновляет выполнение.

Возвращаемые значения кода состояния:

SS$_NORMAL - программа системного обслуживания успешно завершилась;

SS$_ILLEFC - указан неверный номер флага события;

SS$_UNASEFC - процесс не связан с кластером, содержащим указанный флаг события.

13.118. ¤WAKE - пробудить процесс от спячки

Программа системного обслуживания ¤WAKE активизирует процесс, который перевел себя в состояние спячки при помощи программы системного обслуживания "спячка" (¤HIBER).

Формат:

SYS¤WAKE [PIDADR],[PRCNAM]

Аргументы:

PIDADR

Использование в МОС ВП: PROCESS_ID

Тип: длинное слово (без знака)

Доступ: модификация

Механизм: по ссылке

Идентификатор процесса (PID), который требуется пробудить. Аргумент PIDADR - это адрес длинного слова, содержащего PID.

PRCNAM

Использование в МОС ВП: PROCESS_NAME

Тип: строка текста

Доступ: только чтение

Механизм: по дескриптору - дескриптор строки с фиксированной длиной

Имя процесса, который требуется пробудить. Аргумент PRCNAM - это адрес дескриптора символьной строки, указывающего на строку имени процесса длиной от 1 до 15 символов.

Имя процесса неявно квалифицируется номером группы UIC вызывающего процесса. Поэтому аргумент PRCNAM можно использовать только в тех случаях, когда пробуждаемый процесс

находится в той же группе UIC, что и вызывающий процесс. Для пробуждения процесса в другой группе UIC необходимо использовать аргумент PIDADR.

Если не заданы ни аргумент PIDADR, ни аргумент PRCNAM, то запрос на пробуждение выдается для вызывающего процесса.

Описание:

В зависимости от конкретной операции использование программы $WAKE может потребовать наличия у вызывающего процесса одной из следующих привилегий:

1) привилегия GROUP нужна для того, чтобы пробудить другой процесс в той же группе UIC, если данный процесс имеет UIC, отличный от IC вызывающего процесса;

2) привилегия WORLD требуется для пробуждения любого другого процесса в системе.

Если для процесса, который в текущий момент не находится в состоянии спячки, выдан один или несколько запросов на пробуждение, то последующий запрос на спячку сразу же завершится, то есть, процесс не будет переведен в спячку. Для выставленных запросов на пробуждение счетчик не ведется.

Процесс, находящийся в спячке, можно также пробудить при помощи программы системного обслуживания "запланировать пробуждение" ($SCHDWK).

Возвращаемые значения кода состояния:

SS$NORMAL - успешное завершение;

SS$ACCVIO - вызывающая программа не может прочитать строку имени процесса или дескриптор строки; либо не может записать идентификатор процесса;

SS$IVLOGNAM - заданная строка имени процесса имеет нулевую длину или длину более 15 символов;

SS$NONEXPR - предупреждение. Указанный процесс не существует, либо задан неверный идентификатор процесса;

SS$NOPRIV - у процесса нет привилегии на пробуждение указанного процесса.

13.119. $WFLAND - ждать логического "и" флагов событий

Программа системного обслуживания $WFLAND позволяет процессу задавать группу флагов событий, которые он будет ожидать. Процесс переходит в состояние ожидания до тех пор, пока не будут установлены все указанные флаги событий. В данный момент программа $WFLAND возвращает управление вызывающему процессу и выполнение возобновляется.

Формат:

SYS$WFLAND EFN,MASK

Аргументы:

EFN

Использование в МОС ВП: EF_NUMBER

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Номер любого флага события в кластере флагов событий, который должен использоваться программой. Аргумент EFN является длинным словом, содержащим данный номер. Указание какого-то флага события в кластере служит для идентификации данного кластера флагов событий.

Имеется два кластера локальных флагов событий: кластер 0 и кластер 1. Кластер 0 содержит флаги событий от 0 до 31, кластер 1 содержит флаги событий от 32 до 63.

Имеется два кластера общих флагов событий: кластер 2 и кластер 3. Кластер 2 содержит флаги событий от 64 до 95, а кластер 3 содержит флаги событий от 96 до 127.

MASK

Использование: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Флаги событий, которых ожидает процесс. Аргумент MASK является длинным словом, содержащим вектор битов, где каждый бит, если он установлен, указывает соответствующий флаг

события, который ожидает процесс.

Описание:

Состояние ожидания, вызванное данной программой системного обслуживания, может быть прервано асинхронным системным прерыванием (AST), если режим доступа, с которым выполняется AST, равен или является более привилегированным, чем режим доступа процесса, вызвавшего программу ¤WFLAND и процессу разрешены прерывания в данном режиме доступа.

Если состояние ожидания прерывается по AST, то после того как программа обработки AST завершает выполнение, операционная система МОС ЭП повторяет запрос программы ¤WFLAND для процесса. В данной точке, если флаги событий установлены, то процесс возобновляет выполнение.

Возвращаемые значения кода состояния:

SS¤NORMAL - програма системного обслуживания успешно завершилась;

SS¤ILLEFC - указан неверный номер флага события;

SS¤UNASEFC - процесс не связан с кластером, содержащим указанный флаг события.

13.120. ¤WFLOR - ждать логического "или" флагов событий

Программа системного обслуживания ¤WFLOR позволяет процессу задавать группу флагов событий, которые он хочет ожидать. Процесс переводится в состояние ожидания до тех пор, пока не будет установлен хотя бы один из указанных

флагов событий, после чего программа $WFLOR возвращает управление вызывающему процессу и выполнение возобновляется.

Формат:

SYS$WFLOR EFN,MASK

Аргументы:

EFN

Использование в МОС ВП: EF_NUMBER

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Номер любого флага события в кластере флагов событий, который должен использоваться данной программой. Аргумент EFN является длинным словом, содержащим данный номер. Указание какого-то флага события в кластере служит для идентификации данного кластера флагов событий.

Имеется два кластера локальных флагов событий: кластер 0 и кластер 1. Кластер 0 содержит флаги событий от 0 до 31, кластер 1 содержит флаги событий от 32 до 63.

Имеется два кластера общих флагов событий: кластер 2 и кластер 3. Кластер 2 содержит флаги событий от 64 до 95, а кластер 3 содержит флаги событий от 96 до 127.

MASK

Использование в МОС ВП: MASK_LONGWORD

Тип: длинное слово (без знака)

Доступ: только чтение

Механизм: по значению

Флаги событий, которых ожидает процесс. Аргумент MASK является длинным словом, содержащим вектор битов, где каждый бит, если о установлен, указывает соответствующий флаг события, который ожидает процесс.

Описание:

Состояние ожидания, вызванное данной программой системного обслуживания может быть прервано асинхронным системным прерыванием (AST), если режим доступа, с которым выполняется AST, равен или является более привилегированным, чем режим доступа процесса, вызвавшего программу ¤WFLOR и процессу разрешены прерывания в данном режиме доступа.

Если состояние ожидания прервается прерыванием AST, то после того, как программа обработки AST завершает выполнение, операционная система МОС ЭП повторяет запрос программы ¤WFLOR для процесса. В данной точке, если флаг события установлен, то процесс возобновляет выполнение.

Возвращаемые значения кода состояния:

SS¤NORMAL - программа системного обслуживания успешно завершилась;

SS¤ILLEFC - указан неверный флаг события;

SS¤UNASEFC - процесс не связан с кластером, содержащим указанный флаг события.

Перечень ссылочных документов

1. Операционная система МОС ВП. Система управления данными СУД-32. Программирование на языке макроассемблер. Руководство программиста 25.00152-01 33 07

2. Операционная система МОС ВП. Подсистема системного сервиса. Управление вводом-выводом. Руководство программиста 25.00152-01 33 03

3. Операционная система МОС ВП. Система программирования на языке макроассемблер. Описание языка 25.00152-01 35 01

4. Операционная система МОС ВП. Система программирования на языке макроассемблер. Руководство программиста 25.00152-01 33 01

5. Операционная система МОС ВП. Архитектура и система машинных инструкций. Справочный материал 25.00152-01 97 01-1

6. Операционная система МОС ВП. Архитектура и система машинных инструкций. Справочный материал 25.00152-01 97 01-2

7. Операционная система МОС ВП. Подсистема управления. Права доступа. Справочный материал 25.00152-01 97 15-1

8. Операционная система МОС ВП. Подсистема управления. Права доступа. Справочный материал 25.00152-01 97 15-2

9. Операционная система МОС ВП. Подсистема управления. Сообщения системы и действия по восстановлению. Справочный материал 25.00152-01 97 02-1

10. Операционная система МОС ВП. Подсистема управления. Сообщения системы и действия по восстановлению. Справочный материал 25.00152-01 97 02-2

11. Операционная система МОС ВП. Описание применения 25.00152-01 31 01

12. Операционная система МОС ВП. Подсистема исполнительной библиотеки. Руководство программиста 25.00152-01 33 09-1

13. Операционная система МОС ВП. Подсистема исполнительной библиотеки. Руководство программиста 25.00152-01 33 09-2

14. Операционная система МОС ВП. Подсистема исполнительной библиотеки. Руководство программиста 25.00152-01 33 09-3

15. Операционная система МОС ВП. Подсистема исполнительной библиотеки. Руководство программиста 25.00152-01 33 09-4

16. Операционная система МОС ВП. Подсистема исполнительной библиотеки. Руководство программиста 25.00152-01 33 09-5

17. Операционная система МОС ВП. Подсистема разработки программ. Программа компоновщик. Руководство программиста 25.00152-01 33 04

18. Операционная система МОС ВП. Подсистема системного

программиста. Управление системой. Справочный материал

25.00152-01 97 10

Лист регистрации изменений

Номера листов (страниц)					Всего листов (страниц) в докум.	N документа	Входящий N сопроводительного докум. и дата	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулированных					

Министерство приборостроения, средств автоматизации

и систем управления

ОПЕРАЦИОННАЯ СИСТЕМА МОС ВП

Подсистема системного сервиса

Программы системного обслуживания

Справочный материал

Лист утверждения

26.00152-01 97 06-ЛУ

Представители предприятия-разработчика

Руководитель темы
Зам.директора

_____ ХРУДЕВ С.Н.
"___"_____ 1987 г.

Ответственный исполнитель темы
Зав. отделением

_____ ОСТАПЕНКО Г.П.
"___"_____ 1987 г.

Зав. лабораторией

_____ АКСЕНОВ А.В.
"___"_____ 1987 г.

М.н.с.

_____ ПЕРОВ Г.П.
"___"_____ 1987 г.

Техник

_____ СМИРНОВА Т.И.
"___"_____ 1987 г.

Нормоконтролер

_____ СКВОРЦОВ В.А.
"___"_____ 1987 г.

1987

Перв.примен.
26.00152-01

Литера