ОПЕРАЦИОННАЯ СИСТЕМА ОСРВМ СМ ЭВМ

СПРАВОЧНОЕ
ИЗДАНИЕ

Под редакцией Г. А. Е г о р о в а
2-е издание,
переработанное и дополненное

«ФИНАНСЫ И СТАТИСТИКА»
СП «СОВАМИНКО»
АГЕНТСТВО «КОМПЬЮТЕРПРЕСС»
МОСКВА
1990

ББК 32.973
О-76

УДК 681.3.066

Авторы: *Г. А. ЕГОРОВ, В. Л. КАРОЛЬ, И. С. МОСТОВ, О. Б. МАХЛИН, Л. Н. СТОЛЯР, В. А. ШАПОШНИКОВ, В. И. ШЯУДКУЛИС, А. С. ХОЛМЯНСКИЙ*
Рецензент: канд. техн. наук *В. К. КОНДРАТЬЕВ*

Операционная система ОСРВМ СМ ЭВМ: Справ. изд./
О-76 Г. А. Егоров, В. Л. Кароль, И. С. Мостов и др.; Под ред. Г.А.Егорова. — 2-е изд., перераб. и доп. — М.: Финансы и статистика, 1990. — 303 с: ил. ISBN 5-279-00363-8.

Книга является справочным руководством по одной из наиболее широко используемых операционных систем ОСРВМ для СМ ЭВМ. По сравнению с первым изданием (1986 г.) второе издание книги дополнено разделами о системе управления данными, диалоговом командном языке, универсальном текстовом редакторе, программных средствах контроля и управления системой.

Для системных программистов-пользователей СМ ЭВМ, инженеров и научных работников, занимающихся разработкой систем управления на базе СМ ЭВМ.

$\text{О}\frac{2405000000-040}{010\,(\,01\,)-90}115-90$ **ББК** 32.973

ISBN 5-279-00363-8

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Многофункциональная операционная система реального времени (ОСРВМ) является базовой операционной системой для новой модели СМ ЭВМ — вычислительного комплекса (ВК) СМ1425. ОСРВМ может применяться в ВК типа СМ1420, Электроника-79, Электроника-60, но в этом случае не используются новые архитектурные возможности, реализованные в ВК СМ 1425.

Система ОСРВМ предназначена для использования в качестве базовой в различных применениях реального времени и обеспечивает организацию вычислительного процесса в режимах реального времени, разделения времени и пакетной обработки. Она характеризуется развитыми возможностями управления вычислительными ресурсами, имеет широкий набор средств для многопользовательской и многотерминальной работы и большой выбор возможностей для работы с внешними устройствами. В зависимости от требований к составу технических средств и выбираемых функциональных возможностей могут генерироваться различные варианты операционной системы.

ОСРВМ является развитием операционной системы ОСРВ [1]. В отличие от ОСРВ система ОСРВМ имеет ряд существенных различий, направленных на использование и разделение вычислительных ресурсов и на повышение производительности, надежности функционирования вычислительной системы. Эти отличия, прежде всего, связаны с поддержкой новых архитектурных особенностей СМ1425.

В состав ОСРВМ входит широкий набор систем программирования: Макро, Фортран, Кобол, Паскаль, Бейсик. Под управлением системы функционируют программные средства сетевой телеобработки и управления базами данных.

Все описанные функции наряду с широким набором системных и обслуживающих средств, предоставляемых пользователям для реализации проблемных применений, обеспечивают использование ОСРВМ в различных областях.

Настоящая книга является переработанным и дополненным изданием книги «Операционная система ОСРВ СМ ЭВМ». Дополнения, прежде всего, связаны с новыми функциональными особенностями ОСРВМ по сравнению с ОСРВ. По сравнению с предыдущим изданием в книгу внесены новые разделы и главы, описывающие архитектурные особенности ВК СМ1425, диалоговый командный язык, систему управления данными (СУД), программные средства контроля и управления системой, средства генерации системы. Расширено описание ядра операционной системы, директив управляющей программы, построителя задач новыми функциональными возможностями ОСРВМ. Часть разделов первого издания, например описания программы связи с оператором MCR и системы управления файлами FCS, исключены из настоящей книги, так как они входят в состав ОСРВМ без каких-либо изменений.

Излагаемый материал делится на семь глав. В главе 1 рассматриваются архитектурные особенности ВК СМ 1425 и его отличия от ВК СМ1420, а также назначение, структура и функциональные возможности и особенности ОСРВМ по сравнению с ОСРВ.

Глава 2 посвящена описанию директив управляющей программы, обеспечивающих доступ задач к различным ресурсам системы, управление памятью, взаимодействие между задачами.

В главе 3 дано описание системы управления данными (СУД), расширяющей возможности файловой системы ОСРВМ для обработки файлов с последовательной, относительной и индексной организациями.

Глава 4 содержит описание диалогового командного языка (DCL) для связи оператора с системой.

В главе 5 даны основные сведения, необходимые для создания (построения) задач, рассматриваются структура и особенности привилегированных и многопользовательских задач, перекрытий и задач с использованием I/D-пространства.

Глава 6 содержит описание программных средств для управления и контроля ресурсами ОСРВМ и повышения надежности функционирования системы, а глава 7 — сведения, необходимые для работы с обслуживающими программами ОСРВМ.

Предлагаемая книга представляет собой справочное издание и предназначается для системных и проблемных программистов — пользователей СМ ЭВМ, а также для инженеров и научных работников, занимающихся разработкой различных систем управления. Она может быть полезна также аспирантам и студентам вузов.

1. Основные характеристики ОСРВМ

Многофункциональная операционная система реального времени (ОСРВМ) является дальнейшим развитием одной из наиболее распространенных систем для СМ ЭВМ —ОСРВ [1]. ОСРВМ предназначена для применения в качестве базовой операционной системы для вычислительного комплекса (ВК) СМ1425, хотя она может функционировать и в комплексах типа СМ1420, но в этом случае ОСРВМ не использует новые архитектурные возможности, реализованные в ВК СМ1425.

1.1. АРХИТЕКТУРНЫЕ ОСОБЕННОСТИ ВК СМ1425

ВК СМ1425 является новой моделью 16-разрядных СМ ЭВМ и дальнейшим развитием архитектуры СМ1420 [2]. Новый вычислительный комплекс обеспечивает программную совместимость с ВК СМ1420, имеет повышенные показатели производительности и надежности, меньшие габариты, массу и потребляемую мощность.

ВК СМ1425 предназначен для систем научных исследований, автоматизированных систем управления, информационно-справочных и других систем.

Процессор СМ1425, разработанный на базе микропроцессорной сборки, выполняет полный набор команд СМ1420 и ряд дополнительных команд.

В отличие от интерфейса «общая шина» в ВК СМ1420 связь процессора СМ1425 с оперативной памятью и внешними устройствами осуществляется через 22-разрядный магистральный параллельный интерфейс (МПИ).

Интерфейс МПИ реализован на основе магистрали и логических узлов, входящих в каждое подключаемое устройство.

Магистраль состоит из объединенных по функциональному назначению линий обмена информацией, управления обменом, передачи управления, прерывания и вспомогательных линий. Большинство линий магистрали обеспечивает двустороннюю передачу сигналов. Устройства подключаются к линиям параллельно.

Интерфейс МПИ используется процессором и всеми внешними устройствами с разделением во времени и в соответствии с установленным приоритетом. В любой операции обмена всегда участвуют устройства, связанные между собой как задатчик и исполнитель. По отношению к МПИ процессор является устройством с программно изменяемым приоритетом: внешние устройства имеют фиксированные приоритеты.

Принцип построения СМ1425 агрегатный. Он позволяет создавать разнообразные по набору и количеству технических средств конфигурации комплексов. Все функциональные блоки ВК СМ1425 выполнены в виде конструктивно законченных модулей, связанных между собой через МПИ.

ВК СМ1425 отличается от ВК. СМ1420 следующими особенностями:

- тремя режимами работы процессора (режимы ядра, супервизора и пользователя), что обеспечивает большую надежность и гибкость программного обеспечения;
- двумя группами регистров диспетчера памяти (ДП) для каждого из режимов работы процессора, что позволяет увеличить виртуальное адресное пространство задачи до 64 Кслов;
- двумя наборами по 8 регистров общего назначения, что уменьшает накладные расходы операционной системы при диспетчеризации задач и обработке прерываний;
- дополнительными командами, связанными с режимом супервизора, установкой приоритета в PSW и т. п.;
- возможностью блочной передачи данных внешними устройствами, работающими по прямому доступу к оперативной памяти (магнитные ленты и диски);
- аппаратным механизмом программных запросов на прерывание;
- большим объемом хранимых в ПЗУ процессора встроенных тестов;
- реализацией кэш-памяти, т. е. быстродействующей буферной памяти процессора объемом 8 Кбайт, используемой для хранения содержимого ячеек оперативной памяти, обращение к которым непосредственно предшествует текущему состоянию программы.

Организация памяти в СМ1425 — страничная, размер страницы— от 64 до 8192 байт. Каждая страница обеспечивается защитой доступа и возможностью ее перемещения. Реализовано

динамическое преобразование адресов аппаратными и программными средствами. Максимальный объем оперативной памяти — 4 Мбайта.

Диспетчер памяти (ДП) содержит три набора регистров (APR), используемых для преобразования виртуальных адресов в физические в различных режимах процессора. Каждый из наборов APR содержит две группы по 8 пар регистров адреса страницы (PAR) и регистров описания страницы (PDR). Одна группа регистров используется для адресации пространства инструкций (I-пространство), другая — для адресации пространства данных (D-пространство). Благодаря раздельному обращению к пространствам инструкций и данных размер задачи может быть расширен до 64 Кслов. Пространство инструкций используется для выбора инструкций, индексных слов, абсолютных адресов и непосредственных операндов, пространство данных— для всех других обращений.

ВК СМ1425 состоит из базового блока, выполненного в виде небольшой стойки размером 560×200×720 мм. В базовом блоке имеется монтажный блок с посадочными местами для установки восьми блоков элементов (БЭ). Предусмотрена возможность подключения к базовому блоку дополнительного блока, что увеличивает количество посадочных мест.

Для подключения к СМ1425 накопителей на магнитных дисках (НМД) и накопителей на гибких магнитных дисках (НГМД) используется комбинированный контроллер. К контроллеру может быть подключено два НМД типа «винчестер» емкостью 11 Мбайт или 31 Мбайт и два НГМД емкостью 0,8 Мбайта.

Терминалы подключаются с помощью 4-канальных мультиплексоров, имеющих выходы на стыки С2 и ИРПС. Консольный терминал подключается непосредственно к процессору.

Групповой контроллер — комбинированное устройство, обеспечивающее подключение четырех каналов передачи данных:

стык С2 с цепями модемного управления;

стык С2 без цепей модемного управления;

канал с 8-разрядной шиной данных;

канал с 16-разрядной шиной данных.

Обеспечивается возможность подключения к СМ1425 магнитных лент типа СМ5316, СМ5308 или совместимых с ними, а также магнитных лент типа «картридж».

1.2. НАЗНАЧЕНИЕ И СТРУКТУРА ОСРВМ

ОСРВМ предназначена для организации вычислительного процесса в режимах реального времени, разделения времени и в пакетном режиме в различных применениях. Система обеспечивает эффективное использование и разделение вычислительных ресурсов за счет поддержки архитектурных особенностей СМ1425, имеет программную и информационную совместимость с ОСРВ и может быть сгенерирована под различные требования. ОСРВМ по сравнению с ОСРВ имеет ряд отличий, направленных, прежде всего на повышение производительности, надежности функционирования и расширение сфер применения:

- управление выполнением системных и пользовательских задач в трех режимах работы процессора;
- возможность разделения в задачах адресного пространства инструкций и данных;
- размещение вторичного пула, что снижает ограничения на размер системного пула;
- включение вспомогательного управляющего драйвера (ACD), что обеспечивает гибкий механизм для использования в ОСРВМ разных типов терминалов с различными кодовыми таблицами без изменения терминального драйвера;
- возможность использования алфавитно-цифрового представления имен каталогов;
- включение средств кэширования дисков, что обеспечивает повышение производительности системы при обмене данными между дисками и памятью;
- возможность оптимизации обработки очередей ввода-вывода к дисковым устройствам;
- возможность переконфигурации системы при изменении конфигурации комплекса;
- возможность замены дефектных блоков на дисках резервными;
- развитие средств резервирования дисков, учета использования вычислительных ресурсов, пакетной обработки и управления очередями, распределения и уплотнения памяти.

ОСРВМ состоит из следующих компонентов:

- ядра операционной системы;

- набора системных управляющих программ;
- набора системных обслуживающих программ;
- систем программирования и средств создания и отладки задач;
- средств генерации системы;
- программных средств управления системой и повышения надежности функционирования комплекса.

Некоторые из указанных компонентов и программ, входящих в их состав, описаны в справочнике подробно, другие, ввиду ограниченности объема справочника, только упомянуты (их описание можно найти в первом издании справочника [1]).

1.2.1. ЯДРО ОПЕРАЦИОННОЙ СИСТЕМЫ

Компоненты ядра, называемого в ОСРВМ *управляющей программой,* являются резидентными в оперативной памяти.

1.2.2. СИСТЕМНЫЕ УПРАВЛЯЮЩИЕ ПРОГРАММЫ

Набор системных управляющих программ включает средства управления данными, взаимодействия с оператором, управления системой.

К средствам управления данными относятся система управления файлами FCS и система управления данными (СУД). FCS обеспечивает последовательный и прямой доступ к файлам [1]. СУД расширяет возможности файловой системы для обработки файлов с последовательной, относительной и индексной организациями.

Средства взаимодействия оператора с системой включают программу связи с оператором MCR и интерпретатор командных строк DCL, которые обрабатывают команды, вводимые с терминала.

Программы управления системой обеспечивают регистрацию пользователей в системе, протоколирование работы системы, процедуры завершения работы системы, средства управления системным выводом.

1.2.3. СИСТЕМНЫЕ ОБСЛУЖИВАЮЩИЕ ПРОГРАММЫ

В соответствии с функциональным назначением системные обслуживающие программы можно разделить на следующие группы: программы работы с томами и файлами, программы редактирования текстов, программа обслуживания библиотек, программы корректировки объектных модулей и образов задач.

Программы работы с томами и файлами включают:

- программу работы с файлами PIP, обеспечивающую обслуживание каталогов и файлов (копирование, переименование, удаление, распечатку и т. д.);
- программу распечатки содержимого файлов и томов в различных форматах;
- программу преобразования файлов FLX для передачи файлов между томами с файловой структурой ОСРВ, ДОС СМ и РАФОС;
- программу сравнения символьных файлов CMP;
- программу копирования и уплотнения тома DSC, позволяющую выполнить сохранение дискового тома на магнитной ленте или на другом диске с уплотнением файлов с целью устранения фрагментации дискового пространства;
- программу копирования и восстановления файлов BRU для сохранения и восстановления томов, каталогов, файлов или отдельных файлов как монтированных, так и немонтированных дисковых томов;
- программу проверки файловой структуры VFY для проверки файловой структуры на дисках с целью определения и идентификации некорректных файлов, потерянных и свободных блоков;
- программу форматирования дисков FMT;
- программу проверки томов BAD для обнаружения дефектных блоков на дисковом томе с целью их дальнейшего изъятия из использования.

Программы редактирования текстов включают: текстовый редактор EDI для редактирования символьных файлов в строковом режиме, универсальный текстовый редактор EDT для редактирования символьных файлов в строковом и экранном режимах с возможностью использования

функциональной клавиатуры, пакетный редактор SLP для редактирования отдельных строк символьных файлов в пакетном режиме.

Программа обслуживания библиотек LBR обеспечивает возможность создания и модификации макроопределений и объектных модулей, а также универсальных библиотек, содержащих однотипные модули произвольного вида.

Программы корректировки включают: программу корректировки объектных модулей PAT и программу корректировки образов задач ZAP для выполнения модификаций в файлах образов задач на диске.

1.2.4. СРЕДСТВА СОЗДАНИЯ И ОТЛАДКИ ЗАДАЧ

ОСРВМ включает набор трансляторов со следующих языков программирования: макроассемблер, Фортран-77, Кобол, Бейсик, Паскаль.

Макроассемблер имеет полный набор средств символического кодирования инструкций процессора, средств распределения памяти, секционирования программ, а также позволяет использовать как системные макрокоманды, так и макрокоманды, написанные пользователем.

Фортран-77 используется для программирования вычислительных задач, а также задач, работающих в реальном времени.

Кобол ориентирован на создание экономических приложений и является подмножеством языка Кобол-81.

Бейсик — один из наиболее простых языков программирования, использующих как элементарные, так и известные математические выражения для выполнения различных операций.

Паскаль как язык высокого уровня дает возможность использовать принцип структурного программирования при разработке программ.

Оттранслированные задачи компонуются построителем задач ТКВ.

Отладка задач, написанных на макроассемблере, осуществляется с помощью программы-отладчика ODT, компонуемой с задачей на этапе ее построения.

1.2.5. ГЕНЕРАЦИЯ СИСТЕМЫ

Цель генерации системы — создание операционной системы ОСРВМ, настроенной на конфигурации оборудования конкретного комплекса и отвечающей требованиям прикладных задач к ее функциональным возможностям. Полный процесс генерации системы заключается в создании готовой к эксплуатации ОСРВМ из исходных программных компонентов, поставляемых пользователю в виде дистрибутива на магнитных носителях. Процесс состоит из следующих этапов:

- подготовки системы к генерации;
- копирования командной процедуры;
- загрузки сгенерированной системы;
- сохранения сгенерированной системы;
- копирования созданной системы.

При подготовке к генерации системы необходимо определить конфигурацию целевого комплекса, на котором будет функционировать создаваемая ОСРВМ, требования к операционной системе, которые зависят от области применения вычислительной системы и от используемых программных продуктов, например систем программирования высокого уровня, а также выбрать рабочий комплекс, на котором будет выполняться генерация системы. В качестве рабочих комплексов могут быть использованы ВК типа СМ1420, СМ1425.

На следующем этапе выполняется копирование дистрибутивной поставки на магнитный диск достаточного объема. В зависимости от выбранного рабочего комплекса копирование выполняется либо с помощью операционной системы, уже существующей для данного комплекса, либо с помощью автономной программы сохранения и восстановления томов BRU, которая включена в дистрибутивную поставку. Следующие этапы генерации выполняются под управлением той же существующей операционной системы или под управлением базовой ОСРВМ из дистрибутивной поставки.

Выполнение командной процедуры SYSGEN — наиболее трудоемкий и длительный этап, в ходе которого на терминал пользователя распечатываются вопросы о программных возможностях создаваемой системы и конфигурации оборудования целевого комплекса. Полученная из ответов

пользователя информация используется для трансляции, построения и инициализации файла образа системы, драйверов внешних устройств и системных задач. Для сокращения трудоемкости и времени выполнения SYSGEN рекомендуется выполнять генерацию системы на целевом комплексе под управлением базовой ОСРВМ с использованием полнофункциональной управляющей программы, автоконфигуратора и файлов сохраненных ответов. Выбор полнофункциональной управляющей программы обеспечивает автоматическое включение всех программных возможностей ОСРВМ и гарантирует успешную установку стандартных программных продуктов. Все ответы пользователя на вопросы командной процедуры сохраняются в специальных файлах. Более того, можно выбрать выполнение части SYSGEN, так называемого PREPGEN, результатом которого будет только создание файлов сохраненных ответов. После проведения анализа указанных файлов и, если необходимо, их корректировки можно выполнить генерацию без повторных ответов на вопросы.

Вся командная процедура разделена на 9 секций, названия которых отражают основные действия каждой. Процедуры секций обычно выполняются в следующем порядке:

1) выбор возможностей SYSGEN. Определяется использование автоконфигуратора, файлов сохраненных ответов, выполнение секций, начиная с указанной или отдельной секции, либо выполнение PREPGEN;

2) выбор возможностей управляющей программы. Определяется создание полнофункциональной или настроенной пользователем управляющей программы, выбор которой требует ответов пользователя по каждой ее возможности и обеспечивает создание эффективной системы специальных применений;

3) выбор конфигурации внешних устройств. Определяются адреса векторов и регистров команд и состояния (РКС), а также характеристики каждого внешнего устройства целевого комплекса. Использование автоконфигуратора избавляет пользователя от необходимости давать ответы на вопросы данной секции;

4) трансляция управляющей программы и драйверов;

5) построение управляющей программы и драйверов;

6) построение привилегированных задач. Строятся такие системные задачи, как программа связи с оператором MCR, программа монтирования томов MOU, программа загрузки драйверов LOA и др. Выполнение процедур данной секции необходимо, если была построена управляющая программа;

7) построение непривилегированных задач. Строятся системные обслуживающие программы. Выполнение процедур данной секции требуется только в том случае, если пользователю необходимо корректировать указанные задачи;

8) создание файла образа системы. С помощью системной обслуживающей программы VMR из файла образа задачи управляющей программы выполняется инициализация файла образа ОСРВМ в соответствии с данными, установленными в файле SYSVMR;

9) добавление устройств. Данная секция является дополнительной и выполняется только по выбору пользователя для включения загружаемых драйверов с загружаемой базой данных в систему, готовую к эксплуатации.

Следующим этапом выполняется программная загрузка системы в оперативную память по команде BOO. На данном этапе определяется работоспособность системы, полученной после завершения SYSGEN.

В процессе сохранения системы, которое осуществляется с помощью командной «строки SAV/WB, ее образ из оперативной памяти вновь записывается в файл образа системы, но уже в аппаратно-загружаемом формате, т. е. создается готовая к работе ОСРВМ.

На завершающем этапе генерации системы можно выполнить установку программных продуктов, а затем — копирование системного диска в целях его защиты от разрушения информации. В некоторых случаях требуется перенести готовую к работе систему на диск другого типа.

1.2.6. СРЕДСТВА УПРАВЛЕНИЯ И ПОВЫШЕНИЯ НАДЕЖНОСТИ ФУНКЦИОНИРОВАНИЯ

В составе ОСРВМ имеется широкий набор программ, предназначенных для управления и контроля ресурсов ОСРВМ, а также для повышения надежности ее функционирования. (Подробно описание программ данного типа приведено в разд. 6.)

Программа обслуживания файла счетов (ACNT) позволяет администратору системы создавать и поддерживать файл учетных счетов. Каждому пользователю в системе

присваиваются имя, код идентификации (UIC), определяются его права доступа к объектам ОСРВМ. Файл счетов содержит также определение операционной среды, доступной программам пользователя после его регистрации в ОСРВМ.

Подсистема консольного протоколирования состоит из драйвера CODRV и задачи COT..., которые обрабатывают запросы ввода-вывода, направленные на псевдоустройство CO: с целью регистрации сообщений в ОСРВМ. Администратору системы предоставляются возможности сохранять и распечатывать протокол работы ОСРВМ, запускать и останавливать консольное протоколирование и т. д.

Подсистема учета использования ресурсов (ПУИР) позволяет собирать более детальную информацию о функционировании ОСРВМ и каждого пользователя с целью измерения производительности, расчета оплаты за использование ресурсов, поиска узких мест в системе. Информация с функционировании собирается в специальный файл транзакций, который может быть преобразован в текстовый файл для проведения последующего анализа.

Программа управления пулом (PMT) позволяет контролировать использование критического ресурса ОСРВМ — динамического пула памяти. Возможность создания вторичного пула уменьшает вероятность перегрузки ОСРВМ при интенсивном использовании динамических структур данных, однако наличие PMT гарантирует оповещение администратора системы о таких ситуациях.

Программа управления конфигурацией (CON) является неотъемлемой частью ОСРВМ, в функции которой входит установка автономного и комплексного режимов внешних устройств, изменение адреса регистра контроля и состояния, адреса вектора прерывания, состояния контроллеров. Программа CON позволяет изолировать неработоспособное оборудование, чтобы оно не влияло на работу системы в целом.

Программа отображения состояния системы (RMD) позволяет в полуграфической форме получить суммарное состояние ряда важных подсистем ОСРВМ, таких, как занятость оперативной памяти, состояние динамического пула, статистика обращений к подсистеме ввода-вывода, кэширование диска, получение списка активных задач, контроль за состоянием отдельной задачи.

Достижение надежности в ОСРВМ обеспечивается применением двух программных компонентов:

программы проверки работоспособности оборудования (ПРОРАБ), которая позволяет имитировать рабочую нагрузку на внешних устройствах (дисках, магнитных лентах, терминалах) и собирать статистику ошибок. Полученные данные дают возможность в неавтономном режиме выполнять исследования работоспособности как отдельных устройств и носителей информации, так и проверять их параллельное функционирование в комплексе;

подсистемы резервирования дисков, которая обеспечивает дублирование информации на дисковых накопителях, образующих резервную пару. При выходе из строя или при превышении допустимого уровня ошибок на первичном накопителе подсистема резервирования автоматически переключает программы на работу со вторым накопителем, что позволяет уменьшить время, в течение которого система является неработоспособной.

Программа RCT, которую также можно отнести к средствам повышения надежности работы системы, обеспечивает автоматическую замену дефектных блоков из числа резервных на дисках, обслуживаемых протоколом управления массовой памятью (ПУМП). При этом пользователь практически не замечает ошибок на диске (до известного предела, определяемого числом резервных блоков на носителе).

В ОСРВМ предусмотрена возможность создания дискового кэша в оперативной памяти, когда наиболее часто используемые блоки диска представляются их копией в памяти. При этом кэширование диска может быть задано для пяти типов операций: ввода-вывода виртуальных, логических блоков, каталогов, предварительного чтения данных и загрузки перекрытий. Кэширование диска позволяет уменьшить время доступа к данным и нагрузку на диск, что благоприятно влияет на его показатели функционирования.

В состав ОСРВМ включены средства, позволяющие выполнить передачу файлов и эмуляцию терминалов между двумя узлами. Данные средства не заменяют сетевого программного обеспечения, но предоставляют возможность взаимодействия двух узлов на уровне обмена файлами без существенных накладных расходов системы.

1.3. ВОЗМОЖНОСТИ УПРАВЛЯЮЩЕЙ ПРОГРАММЫ

Управляющая программа ОСРВМ реализует следующие функции: управление оперативной памятью, управление задачами, обслуживание запросов задач к управляющей программе, обслуживание прерываний.

Единицей выполняемой работы в системе является задача, которая определяется следующими характеристиками: подчиненность, выгружаемость, привилегированность, приоритет и др. Характеристики задачи хранятся в заголовке задачи и блоке управления задачи ТСВ, задаются на этапе построения задачи и могут изменяться при ее установке.

1.3.1. УПРАВЛЕНИЕ ОПЕРАТИВНОЙ ПАМЯТЬЮ

В оперативной памяти размещаются управляющая программа, различные структуры данных, описывающие состояние системы, и задачи. Задачи выполняются в непрерывных областях оперативной памяти, называемых разделами. Каждый раздел определяется следующими характеристиками: имя, тип, размер и базовый адрес.

В ОСРВМ задачи могут выполняться только в системно-управляемых разделах. Память в таких разделах распределяется динамически, а число размещаемых задач ограничивается только размером раздела. Система может перемещать задачи внутри системно-управляемого раздела с целью устранения фрагментации памяти.

Вторым типом разделов в ОСРВМ является раздел регистров устройств, используемых привилегированными задачами для доступа к внешней странице памяти.

В ОСРВМ дополнительно предусмотрен раздел вторичного пула, используемый для расширения системной динамической памяти.

Управление оперативной памятью базируется на понятиях «виртуальное» и «логическое» адресные пространства.

Виртуальное адресное пространство (ВАП) представляет пространство виртуальных адресов, доступных для задачи. В обычных задачах без разделения пространств инструкций и данных ВАП ограничивается размером 32 Кслов. Для задач, использующих разделение пространств инструкций и данных (в дальнейшем I/D-задач) ВАП расширяется до 64 Кслов.

Диспетчер памяти (ДП) выполняет отображение ВАП в физическую память с помощью регистров APR, каждый из которых позволяет отобразить от 32 слов до 4 Кслов непрерывных областей виртуальной памяти. Отображение осуществляется только в область физической памяти (называемой логическим адресным пространством — ЛАП), к которой задача имеет доступ. Механизмом, используемым для отображения ВАП и ЛАП, являются виртуальные адресные окна (в дальнейшем просто окна). Окна могут создаваться статически при построении задачи или динамически при выполнении задачи с помощью директивы CRAW$. Каждое окно отображает непрерывную область виртуальных адресов, начинается на границе 4 Кслов и имеет размер от 32 слов до 32 Кслов.

Для описания окна в заголовке задачи при ее построении создаются блоки окон, которые инициализируются во время установки задачи. Управляющая программа использует информацию из блоков окон для установки регистров APR при отображении задачи.

Логическое адресное пространство задачи включает один или несколько районов. Каждый район является непрерывной областью физической памяти, содержащей инструкции или данные, или областью, зарезервированной для использования одной или более задачами. С помощью директив управления памятью задача может отображать окна на различные районы. Районы могут создаваться при построении задачи или динамически в процессе ее выполнения. Динамические районы позволяют увеличить логическое адресное пространство задачи без изменения ее виртуального пространства. Обычно задача использует одно окно для каждого района, в которое задача может иметь отображение (рис. 1).

В ОСРВМ различают следующие типы районов в соответствии с их использованием и способом построения:

район задачи — непрерывная область памяти, в которой размещаются заголовок, стек и корневой сегмент памяти;

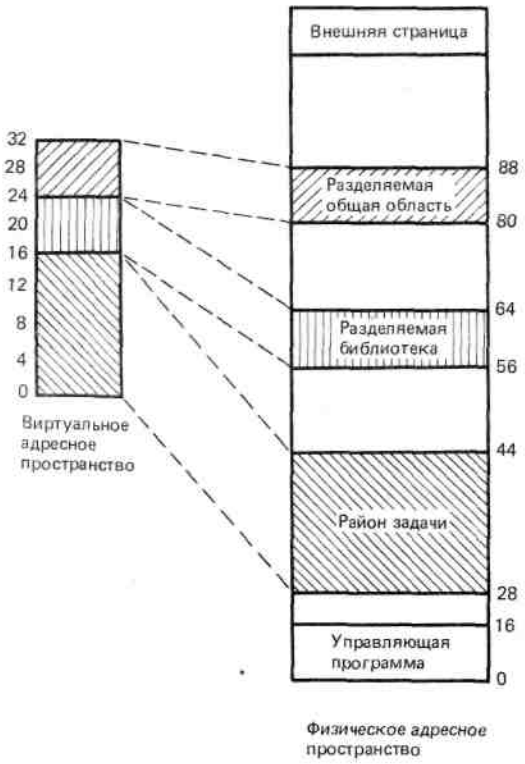

Рис. 1. Отображение виртуального адресного пространства (Кбайт)

разделяемая общая память, используемая для разделения общих данных между задачами;

резидентная библиотека, используемая для разделения библиотечных программ между задачами.

Динамический район создается динамически при выполнении задачи с помощью директивы CRRG$.

В системе без поддержки I/D-пространств (СМ1420) район задачи всегда описывается окном 0. Дополнительно при построении задачи можно задать до 7 окон. Все они нумеруются от 0 до 7.

В системе с поддержкой I/D-пространств (СМ1425) окно 0 описывает корневой сегмент I-пространства, а окно 1—корневой сегмент D-пространства, стек и заголовок задачи. При построении задачи можно дополнительно задать до 14 окон. Все окна нумеруются от 0 до 15.

1.3.2. УПРАВЛЕНИЕ ВЫПОЛНЕНИЕМ ЗАДАЧ

В каждый момент времени на процессоре может выполняться только одна задача. Мультипрограммирование становится возможным потому, что выполнение задачи связано не только с обращениями к процессору. Задача реального времени, вызывая какое-то действие и ожидая его завершения, может не занимать процессор на время ожидания. Поэтому если хотя бы одна задача ждет завершения события (например, завершение операции ввода-вывода), то управляющая программа разрешает другой задаче использовать процессор.

В ОСРВМ управляющая программа координирует выполнение всех задач, находящихся в памяти, чтобы наиболее эффективно использовать ресурсы вычислительной системы. Это достигается за счет особенностей ОСРВМ, которые отражены в следующих понятиях: состояние задачи, приоритет, выгружаемость, круговая диспетчеризация, свопинг, важные события.

Для выполнения в среде ОСРВМ задача должна быть установлена. Установка задачи — это процедура, делающая задачу известной управляющей программе. При установке задачи ОСРВМ помещает ряд ее параметров в блок управления задачей— TCB (имя, размер задачи, адрес на диске, с которого начинается образ задачи, и имя раздела, в котором выполняется задача). TCB включается в каталог установленных задач (STD). Установленная задача определяется как задача, имеющая запись в STD, но еще не загруженная в раздел и не конкурирующая за системные ресурсы. Управляющая программа считает ее бездействующей, пока не будет выдана с терминала или из другой задачи команда, требующая активизации задачи.

Управляющая программа распознает два состояния задачи: бездействие и состояние активности.

Бездействующая задача — это задача, установленная в системе (имеет запись в STD), но не имеющая запроса на выполнение.

Активная задача — это задача, установленная в системе и имеющая запрос на выполнение. Она остается активной, пока не осуществится выход из нее или не произойдет ее аварийное завершение. Тогда она переходит в состояние бездействия. Активная задача может быть в двух состояниях: готова к выполнению и блокирована.

Готовая к выполнению задача конкурирует с другими задачами за процессор на основе приоритетов. Задача с высшим приоритетом получает время на процессоре и становится текущей.

Блокированная задача не способна конкурировать за время процессора по причине занятости необходимых ресурсов.

В системах реального времени различие между бездействующими и активными задачами очень важно. Задача в состоянии бездействия не использует память за пределами управляющей программы, но, когда потребуется ее выполнение, управляющая программа может быстро и эффективно ввести ее в активную конкуренцию за системные ресурсы.

Запись в STD об установленной задаче дает возможность ее быстро активизировать, так как в ней содержатся все параметры, необходимые ОСРВМ для активизации запрошенной задачи. При этом число установленных бездействующих задач может значительно превышать число активных задач. Когда управляющая программа принимает запрос на активизацию бездействующей задачи, она выполняет следующие действия:

распределяет требуемую память;

загружает задачу в раздел (если объем памяти в нем достаточен);

включает задачу в активную конкуренцию за системные ресурсы с другими расположенными в памяти задачами.

Если раздел, в котором задача установлена, полностью занят, и задача не может быть сразу загружена, то она устанавливается в очередь активных задач, упорядоченную по приоритетам.

Активная задача конкурирует за системные ресурсы на основе приоритета, который определяется числом, назначенным задаче при ее построении, при установке или при выполнении. Приоритет находится в пределах от 1 до 250, причем большее число соответствует более высокому приоритету. Приоритет определяет право задачи на ресурсы.

В ОСРВМ выполнение задач реального времени совмещается с менее срочными работами, причем задачи реального времени имеют более высокий приоритет. Такое соглашение принято для того, чтобы управляющая программа отдавала предпочтение задачам реального времени по сравнению с другими задачами.

Конкуренция задач за оперативную память осуществляется с использованием «выгружаемости» задач. *Выгружаемость* — это средство, с помощью которого готовая к выполнению задача, не находящаяся в памяти в настоящее время, может конкурировать за процессор. В некоторых случаях задача не может конкурировать за время процессора, так как раздел, в котором она установлена, полностью занят. Если задача, занимающая раздел, выгружаемая и имеет приоритет ниже, то управляющая программа может выгрузить эту задачу на диск и предоставить освободившийся раздел задаче с более высоким приоритетом. Когда последняя задача завершится, выгруженная задача снова загрузится и продолжит работу с точки, в которой она была прервана и выгружена.

Для выгрузки задачи на диске должно быть достаточное пространство выгрузки, равное размеру задачи. Выгружаемая задача находится в пространстве выгрузки, пока в разделе выполняется задача с более высоким приоритетом.

Пространство выгрузки распределяется либо статически при построении задачи, либо динамически во время выполнения. В ОСРВМ можно применять любой из этих способов выделения пространства выгрузки.

При построении задачи пользователь может затребовать, чтобы пространство выгрузки было размещено в файле образа задачи, который содержит сегменты задачи. В этом случае при выполнении задачи всегда на диске имеется пространство выгрузки для задачи независимо от того, будет ли в действительности управляющая программа выгружать эту задачу.

Динамическое выделение пространства выгрузки позволяет более эффективно использовать диск. Вместо выделения на диске пространства, равного суммарному размеру всех выгружаемых задач, можно создать один файл выгрузки на диске. Этот файл создается по команде оператора независимо от особенностей задачи. Если возможно динамическое выделение пространства выгрузки, то выгружаемость задачи устанавливается при ее построении или при установке. Задача выгружается

управляющей программой в файл выгрузки, если там хватает пространства. Недостатком динамического распределения пространства выгрузки является то, что в файле выгрузки может не оказаться свободного пространства для выгружаемой задачи.

Управляющая программа распределяет ресурсы вычислительной системы, прежде всего время процессора и объем памяти, на основе приоритетов. В ОСРВМ возможны различные режимы диспетчеризации. Два из них — *круговая диспетчеризация* и *свопинг* — влияют на распределение ресурсов, когда многие активные задачи имеют равные приоритеты.

Если несколько находящихся в памяти задач имеют одинаковые приоритеты, управляющая программа стремится отдать время процессора тем задачам, которые оказываются первыми в STD (записи в STD для задач с равными приоритетами обычно появляются в порядке, в котором задачи были установлены). Чтобы сделать такие задачи равноправными, в ОСРВМ используется круговая диспетчеризация. При круговой диспетчеризации периодически просматриваются активные задачи и на каждом уровне приоритетов выполняется циклическое продвижение задач к началу очереди: самая первая задача среди равноприоритетных становится последней.

Иная проблема возникает, когда несколько активных задач с равными приоритетами конкурируют за раздел. Обычно задача не может заставить управляющую программу выгрузить из раздела другую задачу с тем же или более высоким приоритетом.

Свопинг позволяет задачам с равными приоритетами выполнения выгружать друг друга из раздела. Когда задача начинает выполняться, управляющая программа добавляет приоритет свопинга к приоритету выполнения задачи. Во время выполнения задачи управляющая программа постоянно снижает приоритет свопинга, который в конечном счете принимает отрицательное значение. Если сумма уменьшенного приоритета свопинга и приоритета, с которым выполняется задача, меньше, чем приоритет конкурирующей за раздел задачи, то управляющая программа выгружает выполняемую задачу, чтобы отдать пространство раздела конкурирующей задаче.

Управляющая программа затем помещает выгруженную задачу в конец очереди активных задач, конкурирующих за память на этом уровне приоритета. Приоритет свопинга не влияет ни на распределение времени процессора, ни на диспетчеризацию ввода-вывода.

Некоторые ситуации, называемые *важными событиями,* заставляют управляющую программу «просматривать» очередь активных задач и выбирать готовую к выполнению задачу с высшим приоритетом. К важным событиям относятся: завершение ввода-вывода, завершение задачи, выполнение одной или ряда системных директив, выданных задачей, истечение интервала времени при круговой диспетчеризации.

1.3.3. ФУНКЦИИ СИСТЕМНЫХ ДИРЕКТИВ

Системная директива — это запрос из задачи к управляющей программе на выполнение системной функции. Пользовательские задачи используют системные директивы для обмена данными, для управления выполнением и взаимодействием задач. Большинство важных событий связано прямо или косвенно с выполнением системных директив. Системные директивы являются частью управляющей программы ОСРВМ и, явно или неявно, они влияют на то, каким образом управляющая программа распределяет системные ресурсы между активными задачами.

Системные директивы позволяют задачам выполнять следующие функции:

- •получать информацию о системе и о задаче;
- •отменять временной интервал;
- •выполнять операцию ввода-вывода;
- •выполнять действия с логическим и/или виртуальным адресным пространством задачи;
- •приостанавливать и возобновлять выполнение задачи;
- •запрашивать запуск другой задачи;
- •успешно или аварийно завершать выполнение задачи;
- •порождать одну задачу из другой, обеспечивая взаимосвязь между ними; получать информацию об обеих задачах.

Системные директивы позволяют задачам использовать некоторые основные средства ОСРВМ: флаги событий, системные прерывания, расширенное логическое адресное пространство.

Ожидание некоторого события в системе и в задачах может быть связано со значением флага события. Задачи могут использовать флаги событий для синхронизации своей работы и взаимодействия

с другими задачами. Когда происходит важное событие, управляющая программа устанавливает флаг события. Задача может опрашивать состояние флага и тем самым определять, произошло ли соответствующее событие. Задача может сама выдать системную директиву установки флага.

В ОСРВМ имеется 96 флагов событий. Каждый из них имеет соответствующий номер. Первые 32 флага являются *локальными* для каждой задачи и устанавливаются или сбрасываются в результате действия самой задачи. Следующие 32 флага, являясь общими для всех задач, называются *общими.* Общие флаги могут быть установлены или сброшены в результате действий любой задачи. Последние восемь флагов в каждой группе — локальные флаги 25—32, общие флаги 57—64 — зарезервированы для управляющей программы. Флаги третьей группы (65—96) называются групповыми *глобальными* флагами. Эти флаги используются в тех же случаях, что и общие флаги, с той разницей, что ими могут пользоваться лишь задачи, выполняющиеся под кодом идентификации с номером группы, для которой эти флаги созданы.

1.3.4. ОБСЛУЖИВАНИЕ ПРЕРЫВАНИЙ

В системе реального времени гибкость, быстродействие и разнообразие средств реакции на внешние события имеет решающее значение. Полностью возможности быстродействия оборудования могут использоваться программой, выполняющейся на машине без операционной системы. В этом случае программа сама должна обеспечить реакцию на синхронные и асинхронные прерывания.

С и н х р о н н ы е прерывания вызываются самой программой (например, резервная инструкция, EMT и др.) в отличие от асинхронных, вызванных причинами, внешними по отношению к выполнению программы (например, прерывание от таймера или от устройства ввода-вывода).

Системные средства обслуживания прерываний дают возможность передать управление для обработки прерываний в указанную точку задачи пользователя, когда происходит определенное событие. Они позволяют задачам реагировать на синхронные и асинхронные события. В ОСРВМ различают два типа системных прерываний: синхронные (SST), асинхронные (AST). Синхронные системные прерывания всегда возникают в одной и той же точке программы, если повторить ее выполнение. Задача может иметь программу реакции на все виды синхронных прерываний процессора, за исключением инструкции EMT с кодом 377, зарезервированной для обращений к системе.

А с и н х р о н н ы е прерывания возникают асинхронно к выполнению задачи. В ОСРВМ имеется широкий набор возможностей обработки асинхронных прерываний. Прерывание обслуживается системными средствами или задачей пользователя. На системном уровне прерывание может обслуживаться драйвером устройства или задачей с использованием директивы CINT$. Прерывание может обслуживаться на высшем, седьмом, приоритете процессора (т. е. в режиме запрета других прерываний), на приоритете устройства или на нулевом приоритете процессора. В ОСРВМ пользователям предоставляется возможность разработки и включения в систему дополнительных драйверов внешних устройств.

2. Директивы управляющей программы

2.1. ВВЕДЕНИЕ В УПРАВЛЯЮЩУЮ ПРОГРАММУ

Ядром операционной системы ОСРВМ является управляющая программа, которая организует доступ задач к ресурсам системы, поддерживает соответствие между виртуальными, логическими и физическими адресными пространствами, исполняет программные запросы задач к системе, обеспечивает взаимодействие между выполняющимися задачами.

Все программные вызовы, называемые *директивами управляющей программы,* реализуются выполнением инструкции EMT 377.

Данные, указанные в директиве, передаются системе через блок параметров директивы DPB, младший байт первого слова которого содержит код идентификации директивы (нечетное число), следующий за ним байт содержит количество слов в блоке, а остальные байты содержат параметры, относящиеся к данной директиве. В результате обработки директивы система формирует слово состояния директивы DSW со значением +1 при успешном завершении директивы и с отрицательным значением числа (код аварийного завершения директивы), если директива отвергнута.

Дополнительно директива (кроме директив завершения задачи) возвращает управление следующей за ней инструкции с установкой кода условия С: 0 — если директива выполнена успешно, или 1 — если директива отвергнута.

2.2. СОГЛАШЕНИЯ ПО ОПИСАНИЮ ФОРМАТОВ И ВЫЗОВОВ ПОДПРОГРАММЫ

Обращение к директивам в программе, написанной на языке макроассемблер, обеспечивается использованием соответствующих макровызовов. Соответствующие директивам макроопределения содержатся в макробиблиотеке LB: [1, 1] OCMAC.SML.

Для обращения к этим макроопределениям необходимо издать директиву языка макроассемблер .MCALL. Аргументами директивы .MCALL являются имена макрокоманд, используемых в программе пользователя.

Имена макрокоманд содержат не более четырех символов, за которыми следует знак $, а иногда — еще один символ. Добавление последнего символа или его отсутствие определяет, какая из трех возможных форм директивы будет использована.

Если необязательный символ отсутствует ($ — форма директивы), формируется только DPB, который размещается в точке появления макровызова и не содержит машинных инструкций. Эта форма обычно используется в сочетании с макрокомандой DIRS.

Если необязательным символом служит буква S ($S – форма), формируются инструкции засылки блока DPB в стек и выполняется инструкция EMT 377.

Если необязательным символом служит буква С ($С – форма), то адрес ДРВ засылается в стек и выполняется инструкция EMT 377. Блок DPB создается в отдельной программной секции с именем $DPB$$. В исходной программной секции генерируется команда засылки адреса DPB в стек EMT 377.

Чтобы обеспечить возврат в исходную программную секцию, пользователь должен указать имя этой секции непосредственно после параметров самой директивы. Если имя секции не указано, предполагается неименованная секция.

Макрокоманда DIR$ позволяет обращаться к директиве, блок DPB которой создан с использованием формы соответствующей макрокоманды.

Макрокоманда DIR$ генерирует обращение к управляющей программе, передавая ей адрес ранее созданного блока DPB. Макрокоманда DIR$ засылает адрес блока DPB в стек и генерирует инструкцию EMT 377.

$S и $C – формы макрокоманд и макрокоманда DIR$ допускают необязательный последний аргумент, который указывает адрес пользовательской программы обработки ошибок директивы. Если он задан, то генерируется последовательность инструкций:

```
BCC .+N
JSR PC,B,
```

где N — смещение в словах до инструкции, следующей после этих двух;
B — адрес подпрограммы обработки ошибок.

Большинство макроопределений в $ –, $S – и $C – формах порождают символы, определяющие величины смещений в байтах от начала блока до соответствующего его элемента.

Если в программе определен символ $$$GBL (например, $$$GBL = 0), указанные формы не создают блок DPB, а генерируют глобальные символы смещений.

При программировании задач на языке Фортран используется набор подпрограмм, выполняющих функции вызова системных директив. Эти подпрограммы содержатся в библиотеке объектных модулей, используемой при построении задачи, написанной на языке Фортран. Для обращения к любой из них необходимо в программе указать оператор CALL с именем соответствующей подпрограммы. Для передачи в подпрограмму необходимых аргументов должны соблюдаться следующие правила.

Аргумент (имя задачи) определяется переменной типа REAL и кодируется в коде RADIX-50. Эта переменная может быть определена на этапе компиляции оператором DATA.

Аргумент (целый массив), содержащий адреса переменных и массивов, заполняется посредством вызова подпрограммы GETADR. Оператор CALL GETADR (M, [A1], [A2], ... [AN]) позволяет записать адреса аргументов A1 и т. д. в последовательные элементы массива M. Если аргумент опущен, то запись в соответствующий элемент массива не производится.

Каждый вызов директивы на языке Фортран может включать необязательный аргумент — код завершения директивы (КЗД). Его значения соответствуют значениям кода завершения директивы, возвращаемым в слове состояния директивы $DSW.

2.3. ГРУППЫ ДИРЕКТИВ

Директивы группируются по следующим функциям:

- директивы управления выполнением задач;
- директивы управления состоянием задач;
- информационные директивы;
- директивы, связанные с событиями;
- директивы, связанные с обработкой прерываний;
- директивы ввода-вывода и межзадачных связей;
- директивы управления памятью;
- директивы порождения задач;
- директивы, использующие дополнительные возможности;
- директивы, предназначенные для интерпретаторов командных строк CLI.

Директивы управляющей программы рассматриваются далее по указанным группам в алфавитном порядке. При описании аргументов директив используются следующие основные обозначения:

TSK — имя задачи;
IDS — переменная типа INTEGER∗2 для кода завершения директивы;
ERR — адрес подпрограммы обработки ошибок;
LUN — логический номер устройства;
MOD — модификатор логического имени в таблице;
PRI — приоритет;
DEV — имя устройства;
UNT — номер устройства;
RDB — адрес блока описания режима RDB;
WDB — адрес блока описания окна WDB;
VEC — адрес вектора прерываний;
BASE — виртуальный базовый адрес для отображения программы обработки прерываний ISR и программы разрешения/запрещения прерываний при помощи системной пары регистров диспетчера памяти APR5;
ISR — виртуальный адрес ISR или 0 для отключения задачи от вектора прерываний;
EDIR — виртуальный адрес программы разрешения/запрещения прерываний;
AST — адрес подпрограммы обработки асинхронного системного прерывания (AST-прерывание);
EFN — номер флага события;
TBNUM — номер таблицы логических имен;
LUS — логическое имя;
LUSSZ — размер логического имени в байтах;
ENS — эквивалентное имя;
ENSSZ — размер области данных эквивалентного имени в байтах;
STATUS — статус логического имени;
TNAME — имя задачи-потомка;
LAST — адрес AST-подпрограммы, которая вызывается при завершении задачи-потомка или при выдаче ею

состояния;
ESB — адрес блока состояния, который заполняется при завершении задачи-потомка или при выдаче ею состояния;
NAME — имя района;
GROUP — номер группы пользователей, для которой создаются групповые глобальные флаги;
IAST — адрес AST-подпрограммы, обслуживающей запросы на ввод задачи-потомка;
OAST — адрес AST-подпрограммы, обслуживающей запросы на вывод задачи-потомка;
AAST — адрес AST-подпрограммы, в которой задача-родитель может получить извещение об успешном завершении присоединения или отсоединения виртуального терминала;
MBN — максимальный размер буфера (в байтах) для запросов ввода-вывода;
UNUM — номер виртуального терминала;
INC — число 32-словных блоков, на которое увеличивается или уменьшается размер задачи;
SYM — имя системной характеристики;
RESERV — зарезервированный параметр, который должен быть пустым;
WVEC — адрес таблицы, содержащей несколько блоков WDB. Таблица завершается словом-ограничителем:
PRT — имя раздела;
RID — идентификатор района;
TMG — величина интервала времени;
TUT — единица измерения интервала времени;
MAST — 7- битная маска, каждый бит которой соответствует регистру APR в пространстве данных режима супервизора;
TBMSK — маска запрета, предотвращающая поиск в таблице логических имен;
FUC — код функции ввода-вывода;
ISB — адрес блока состояния ввода-вывода;
PRL — список параметров;
PARENT — имя родителя;
UGC — номер группы UIC задачи;
UMC — номер пользователя группы UIC задачи;
SC — флаговые биты, задающие функции директивы;
OCBAD — адрес блока OCB, передаваемого запрашиваемой задаче;
OPT — имя 4 - словного целого массива, где OPT(1) — первая половина имени раздела в RADIX-50; OPT(2) — вторая половина имени раздела в RADIX-50; OPT(3) — приоритет; OPT(4) — UIC пользователя;
SMG — приращение времени для диспетчеризации задачи;
SNT — единицы задания приращения времени;
RMG — величина интервала передиспетчеризации;
RNT — единицы задания интервала передиспетчеризации;
GRP — номер группы флагов;
MSK — 16-битное слово маски флагов события;
RTBMOD — режим приема данных переменной длины;
ADR — адрес таблицы векторов SST-прерываний;
TBNUM — номер таблицы логических имен.

2.3.1. ДИРЕКТИВЫ УПРАВЛЕНИЯ ВЫПОЛНЕНИЕМ ЗАДАЧ

Директивы управления выполнением задач обеспечивают главным образом запуск и останов задач. Каждая из этих директив (кроме директивы расширения памяти задачи) приводит к изменению памяти задачи, если задача еще не находится в запрашиваемом состоянии.

Директива ABRT$ — аварийно завершить задачу — заканчивает выполнение указанной задачи. Вызов на языке Фортран:

```
CALL ABORT (TSK, [IDS])
```

Макровызов:

```
ABRT$ TSK
```

Директива CSRQ$ —отменить требования запуска задачи по времени — отменяет все требования на запуск по времени указанной задачи, не считаясь с источником требований, т. е. являются ли эти требования результатом директивы RUN$ либо результатом одной из форм команды RUN MCR или DCL.

В системе с многопользовательской защитой непривилегированная задача может издать директиву CSRQ$ только для отмены запросов на запуск тех задач, которые имеют в качестве TI: тот же терминал, что и задача, издающая директиву. Вызов на языке Фортран:

```
CALL CANALL (TSK ,[IDS])
```

Макровызов:

```
CSRQ$ TASK
```

Директива EXIT$S — завершить задачу — заканчивает задачу, издавшую директиву. Рекомендуется $S– форма. Вызов на языке Фортран:

```
STOP
```

или

```
CALL EXIT
```

Макровызов:

```
EXIT$S ERR
```

Директива EXTK$ — расширить память задачи — изменяет размер памяти задачи, издавшей директиву на положительную или отрицательную величину, заданную в 32-словных блоках. Вызов на языке Фортран:

```
CALL EXTTSK ([INC], [IDS])
```

Макровызов:

```
EXTK$  [INC]
```

Директива RQST$ — запустить задачу — активизирует задачу. Задача будет запущена на выполнение в соответствии с приоритетом при условии выделения ей необходимого объема памяти. Директива RQST$ является основным механизмом, используемым задачами для инициации других, установленных в системе, но бездействующих задач. Вызов на языке Фортран:

```
CALL REQUES (TSK, [OPT]  [, IDS])
```

Макровызов:

```
RQST$  TSK,  [PRT],  [PRI]  [,UGC, UMC]
```

Директива RSUM$— возобновить задачу — начинает выполнение задачи, издавшей директиву SPND$ (приостановить задачу). Вызов на языке Фортран:

```
CALL RESUME (TSK, [IDS])
```

Макровызов:

```
RSUM$  TSK
```

Директива SPND$S — приостановить задачу — приостанавливает выполнение задачи, издавшей директиву. Рекомендуется $S-форма.

Задача может приостановить только себя, но не другую задачу. Задача может быть запущена снова директивой RSUM$ (возобновить задачу), командой MCR RES или командой DCL CONTINUE. Вызов на языке Фортран:

```
CALL SUSPND  [(IDS)]
```

Макровызов:

```
SPND$S  [ERR]
```

Директива RUN $ — запустить задачу в указанное время — заставляет систему активизировать задачу в определенный момент времени и, по желанию пользователя, периодически повторять запуск этой задачи. Время диспетчеризации задачи задается при помощи приращения относительно момента издания директивы. Если параметры «SMG», «RMG» и «RNT» опущены, то директива RUN$ подобна директиве RQST$ (запустить задачу) с тем отличием, что, во-первых, задача будет активизирована через один тик от момента издания директивы RUN$ и, во-вторых, система всегда устанавливает в качестве устройства TI: запрошенной задачи устройство CO:. Вызов на языке Фортран:

```
CALL RUN (TSK, [OPT],SMG,SNT, [RMG], [RNT]  [,IDS])
```

или

```
CALL START (TSK,SMG,SNT [,IDS])
```

Макровызов:

```
RUN$ TSK,  [PRT],  [PRI],  [UGC],  [UMC],SMG,SNT [,RMG, RNT]
```

2.3.2. ДИРЕКТИВЫ, УПРАВЛЯЮЩИЕ СОСТОЯНИЕМ ЗАДАЧ

Первая директива этой группы изменяет приоритет выполнения задачи, а две другие влияют на свойство выгружаемости.

Директива ALTP$*—изменить приоритет задачи* — изменяет приоритет выполнения активной задачи, указанной в этой директиве, следующим образом:

1) устанавливает приоритет, указанный в директиве;

2) если в директиве значение приоритета не задано, устанавливает приоритет задачи, равным значению приоритета по умолчанию (равным приоритету, получаемому задачей при установке).

Задача, для которой издана директива ALTP$, должна быть установлена и активна. При завершении задачи управляющая программа устанавливает приоритет задачи, равным приоритету по умолчанию (приоритету, получаемому задачей при ее установке).

В системе с многопользовательской защитой непривилегированная задача может издать директиву ALTP$ только для себя и может установить свой приоритет только меньшим или равным приоритету, заданному при установке. Привилегированная задача может изменить приоритет любой задачи в пределах от 1 до 250. Вызов на языке Фортран:

```
CALL ALTPRI ([TSK], [PRI]  [,IDS])
```

Макровызов:

```
ALTP$  [TSK]  [,PRI]
```

Директива DSCP$*S— запретить выгружаемость задачи* — запрещает выгружаемость задачи, которая была установлена как выгружаемая. Директива DSCP$S воздействует только на задачу, издавшую директиву. Не может быть запрещена выгружаемость другой задачи. Рекомендуется $S-форма директивы. Вызов на языке Фортран:

```
CALL DISCKP([IDS])
```

Макровызов:

```
DSCP$S [ERR]
```

Директива ENCP$*S — разрешить выгружаемость задачи* — возобновляет выгружаемость задачи, установленной как выгружаемая, т. е. прекращает действие директивы DSCP$S. Рекомендуется $S-форма директивы. Вызов на языке Фортран:

```
CALL ENACKP ([IDS])
```

Макровызов:

```
ENCP$S [ERR]
```

2.3.3. ИНФОРМАЦИОННЫЕ ДИРЕКТИВЫ

Информационные директивы передают запрашивающей задаче системную информацию: параметры задачи, состояние консольных переключателей, параметры раздела или района, время дня.

Директива FEAT$ *—проверить указанные системные характеристики* — проверяет, поддерживает ли система те или иные возможности, например, работу с процессором плавающей запятой. Вызов на языке Фортран:

```
CALL FEAT (SYM [,IDS])
```

Макровызов:

```
FEAT$  SYM
```

Директива GDIR$ *—дать информацию о каталоге* — находит строку каталога по умолчанию и передает ее и ее длину в указанный пользователем буфер. Вызов на языке Фортран:

```
CALL GETDDS (MOD,EUS,EUSSZ, [RSIZE]  [,IDS])
```

Макровызов:

```
GDIR$  [MOD],EUS, EUSSZ [,RSIZE]
```

где MOD — модификатор для директивы GDIR$;
RSIZE — адрес буфера, в который передается действительный размер строки каталога по умолчанию.

Директива GPRT$ *— дать параметры раздела* — возвращает информацию о разделе в 3-словный

буфер. Если раздел не указан, то предполагается раздел задачи, издавшей директиву. Вызов на языке Фортран:

```
CALL GETPAR ([PRT], BUF[, IDS])
```

где BUF — 3-словный целый массив для параметров раздела.

Макровызов:

```
GPRT$  [PRT],BUF
```

где BUF — адрес 3-словного буфера для параметров раздела.

Формат буфера:

слово 0 — базовый физический адрес раздела в 32-словных блоках (раздел всегда выравнен на границу 32-словного блока). Так, для раздела, начинающегося с 40000 (8), это слово будет содержать 400 (8);

слово 1 — размер раздела в 32-словных блоках;

слово 2 — слово флагов раздела; это слово устанавливается в 0, если раздел системно-управляемый, и в 1, если это раздел, управляемый пользователем.

Директива GREG$ —дать параметры района — возвращает параметры района в 3-словном буфере. Если район не задан, то подразумевается район задачи, издавшей директиву. Вызов на языке Фортран;

```
CALL GETREG ([RID] , BUF[,IDS])
```

где BUF — 3-словный целый месяц для возвращения параметров района.

Макровызов:

```
GREG$  [RID] , BUF
```

где BUF — адрес 3-словного буфера для возвращаемых параметров района.

Формат буфера:

слово 0 — базовый адрес района в 32-словных блоках (адрес района всегда выравнен на границу 32-словного блока). Например, начало района 1000 (8) передается как 10 (8);

слово 1 — размер района в 32-словных блоках;

слово 2 — слово флагов района; это слово равно 0, если район размещен в системно-управляемом разделе, и равно 1 для района, являющегося разделом, управляемым пользователем.

Директива GSSW$S— дать значение консольных переключателей — получает значение слова консольных переключателей процессора и записывает его в слово состояния директивы задачи (только для ВК типа СМ 1420). Рекомендуется $S-форма директивы. Вызов на языке Фортран:

```
CALL READSW (ISW)
```

где ISW — целая переменная для значения консольных переключателей.

Макровызов:

```
GSSW$S [ERR]
```

Директива GTIM$ — дать параметры времени — заполняет указанный 8-словный буфер значением текущего времени. Все параметры времени возвращаются как двоичные числа. Десятичный диапазон величин приведен ниже в описании формата буфера. Вызов на языке Фортран:

```
CALL GETTIM (IBFP[,IDS])
```

где IBFP — имя целого 8-словного массива для получения параметров времени.

Макровызов:

```
GTIM$  BUF
```

где BUF — адрес 8-словного буфера.

Формат буфера:

слово 0 — год (начиная с 1900);
слово 1 — месяц года (1—12);
слово 2 — день месяца (1—31);
слово 3 — час дня (0—23);
слово 4 — минуты (0—59);
слово 5 — секунды (0—59);
слово 6 — тики (0—49);
слово 7 — число тиков в секунду (50).

Директива GTSK$ —дать параметры задачи — заполняет указанный в директиве 16-словный буфер

параметрами, относящимися к задаче, издавшей директиву. Вызов на языке Фортран:

CALL GETTSK (BUF [,IDS])

где BUF—16-словный целый массив для получения параметров задачи. Макровызов:

GTSK$ BUF

где BUF—адрес 16-словного буфера. Формат буфера:

слово 0 — имя задачи, издавшей директиву (первая половина имени в RADIX·50);
слово 1 — имя задачи, издавшей директиву (вторая половина имени в RADIX·50);
слово 2 — имя раздела (первая половина имени в RADIX·50);
слово 3 — имя раздела (вторая половина имени в RADIX·50);
слово 4 — резерв;
слово 5 — резерв;
слово 6 — приоритет выполнения;
слово 7 — UIC задачи, издавшей директиву; в системе с многопользовательной защитой — UIC задачи по умолчанию;
слово 8 — число логических устройств ввода-вывода (число LUN);
слово 9 — номер модели процессора;
слово 10 — резерв;
слово 11 — адрес таблицы SST – векторов задачи;
слово 12 — размер таблицы SST – векторов задачи в словах;
слово 13 — в системе с диспетчером памяти размер нулевого адресного окна задачи (в байтах); в системах без диспетчера памяти размер раздела задачи (в байтах);
слово 14 — признак системы, в которой выполняется задача (для ОСРВМ — 6);
слово 15 — UIC по защите. В системе с многопользовательской защитой — UIC регистрации пользователя.

Директива TFEA$ *—проверить указанные характеристики задачи* — позволяет проверить наличие указанных характеристик задачи, таких, как состояние задачи. Вызов на языке Фортран:

CALL TFEA (ISYM, IDS)

где ISYM —имя характеристики задачи.

Макровызов:

TFEA$ SYM

где SYM — имя характеристики задачи.

2.3.4. ДИРЕКТИВЫ, СВЯЗАННЫЕ С СОБЫТИЯМИ

Директивы, связанные с событиями и флагами событий, являются средством, предназначенным для синхронизации между задачами и внутри задач. Директивы данной группы позволяют также устанавливать системное время. Необходима осторожность в использовании данных директив, так как ошибки, возникающие в результате неверной синхронизации, часто остаются незаметными и их трудно устранить.

Директива CLEF$ *— очистить флаг* — обнуляет указанный флаг события и передает значение флага перед его чисткой в коде завершения директивы. Вызов на языке Фортран:

CALL CLREF (EFN [,IDS])

Макровызов:

CLEF$ EFN

Директива CMKT$ *— отменить запросы директивы MRKT*$ — отменяет указанные запросы директивы MRKT$ или все запросы директивы MRKT$, изданные данной задачей. Вызов на языке Фортран:

CALL CANMT (EFN [,IDS])

Макровызов:

CMKT$ [EFN], [AST]

Директива CRGF$*—создать групповые глобальные флаги* — создает блок описания глобальных флагов (GFB) и включает его в список имеющихся GFB. Если в момент издания директивы блока для указанной группы пользователей не существует, то такой блок создается, а все флаги в нем обнуляются. Если же в момент издания директивы блок GFB для указанной группы уже существует, то новый блок

GFB не создается, и флаги в имеющемся блоке GFB не изменяются. Если блок GFB отмечен к удалению (по изданной ранее директиве ELGF$), то по директиве CRGF$ в нем очищается бит удаления GS.DEL. Вызов на языке Фортран:

```
CALL CRGF ([GROUP]  [,IDS])
```

Макровызов:

```
CRGF$  [GROUP]
```

Директива DECL$S — объявить важное событие — объявляет важное событие. По директиве DECL$S управляющая программа ищет в списке активных задач задачу с наивысшим приоритетом, готовую к выполнению. Эту директиву следует использовать осторожно, так как ее непродуманное применение приводит к чрезмерному возрастанию непроизводительных расходов системы. Рекомендуется $S-форма директивы. Вызов на языке Фортран:

```
CALL DECLAR ([,IDS])
```

Макровызов:

```
DECL$S  [,ERR]
```

Директива ELGF$ —уничтожить групповые глобальные флаги — помечает групповые глобальные флаги на удаление. Если указанные групповые глобальные флаги не используются ни в одной задаче, т. е. если счетчик доступа к флагам G.CNT в блоке GFB равен нулю, то блок GFB уничтожается — блок исключается из списка блоков GFB, и память, занятая под блок освобождается. Если же есть задачи, использующие указанные групповые глобальные флаги, то управляющая программа только помечает их на удаление — устанавливает бит GS.DEL равным 1. Блок GFB, помеченный на удаление, будет уничтожен после того, как не останется задач, ожидающих эти флаги. Однако если до того, как блок GFB был уничтожен, издана директива CRGF$, то бит GS.DEL очищается. Вызов на языке Фортран:

```
CALL ELGF ([GROUP],  [IDS])
```

Макровызов:

```
ELGF$  [GROUP]
```

Директива EXIF$ —завершить задачу, если не установлен флаг — завершает задачу, издавшую директиву, если указанный флаг события не установлен. Если флаг установлен, то управляющая программа возвращает управление задаче. Вызов на языке Фортран:

```
CALL EXITIF (EFU [,IDS])
```

Макровызов:

```
EXIF$  EFU
```

Директива MRKT$ — объявить важное событие через таймерный интервал — заставляет управляющую программу объявить важное событие через указанный интервал времени. Интервал отсчитывается с момента издания директивы, однако выполнение задачи в течение этого интервала продолжается. Если указан флаг события, то при издании директивы он «обнуляется» и устанавливается в момент объявления важного события. Если указан адрес входа в AST – программу, то в момент объявления важного события для задачи объявляется асинхронное системное прерывание, которое будет обрабатываться этой AST – программой. На входе в AST – программу в стек задачи будут помещены PS и PC задачи в точке прерывания, состояние директивы, слово маски флагов, ожидаемых задачей, номер флага, указанного в директиве. Если ни номер флага события, ни адрес программы обработки AST – прерывания не заданы, важное событие все равно будет объявлено по истечении указанного интервала времени. Вызов на языке Фортран:

```
CALL MARK ([EFN], TMG, TNT [,IDS])
```

Макровызов:

```
MRKT$  [EFN], TMG, TNT [,AST]
```

Директива RDAF$ —считать все флаги — считывает 64 флага события для задачи, издавшей директиву, и записывает их значение в 64-битный (4-словный) буфер. Эта директива не возвращает

значение групповых глобальных флагов (флаг с 65-го по 96-й). Вызов на языке Фортран:

CALL READEF (EFN, [IDS])

Здесь подпрограмма READEF считывает только один указанный флаг и возвращает его значение в IDS.

Макровызов:

RDAF$ BUF

где BUF—адрес 4-словного буфера.

Формат буфера:

слово 0 — локальные флаги задачи с 1 по 16;
слово 1 — локальные флаги задачи с 17 по 32;
слово 2 — общие флаги задачи с 33 по 48;
слово 3 — общие флаги задачи с 49 по 64.

Директива RDEF$ *— считать единичный флаг* — проверяет указанный флаг и передает его значение в DSW. Вызов на языке Фортран:

CALL READEF (EFN[,IDS])

Макровызов:

RDEF$ EFN

Директива RDXF$ *— считать расширенные флаги* — считывает для задачи, издавшей директиву, все локальные, общие, групповые глобальные флаги и записывает их значение в 96-битный (6-словный) буфер. Вызов на языке Фортран:

CALL READEF (EFN, IDS)

Здесь подпрограмма READEF может считать для задачи только один флаг.

Макровызов:

RDXF$ BUF

где BUF — адрес 6-словного буфера.

Формат буфера:

слово 0 — локальные флаги задачи с 1 по 16;
слово 1 — локальные флаги задачи с 17 по 32;
слово 2 — общие флаги задачи с 33 по 48;
слово 3 — общие флаги задачи с 49 по 64;
слово 4 — групповые глобальные флаги задачи с 65 по 80;
слово 5 — групповые глобальные флаги задачи с 81 по 96.

Директива SETF$ *—установить флаг* — устанавливает указанный флаг события и сообщает значение флага, которое было перед установкой. Значение флага возвращается в DSW. Вызов на языке Фортран:

CALL SETEF (EFN [,IDS])

Макровызов:

SETF$ EFN

Директива STIM$ *— установить системное время* — устанавливает системное время в соответствии с заданными параметрами директивы. В зависимости от заданных параметров директива, прежде чем устанавливать системное время, выдает текущее значение времени. Директива STIM$ может быть издана только привилегированной задачей. Вызов на языке Фортран:

CALL SETTIM (IBUFN[,IBUFP[,IDS]])

где IBUFN — 8-словный целый массив, содержащий значение устанавливаемого времени;
IBUFP — 8-словный целый массив, в котором возвращается текущее значение времени — времени, которое было до изменения.

Макровызов:

STIM$ BUFN[,BUFP]

где BUFN — адрес 8-словного буфера, содержащего новое значение времени; BUFP — адрес 8-словного буфера, в котором

возвращается текущее значение времени.

Формат буфера:

слово 0 — год (начиная с 1900);
слово 1 — месяц (1—12);
слово 2 — день (1—N, где N — наибольшее значение дня для указанного месяца года);
слово 3 — часы (0—23);
слово 4 — минуты (0—59);
слово 5 — секунды (0—59);
слово 6 — тики (0—N, где N — частота системного таймера минус 1). Если следующий параметр (число тиков в секунде) опущен, то этот параметр игнорируется;
слово 7 — число тиков в секунде (либо должен быть опущен, либо должен соответствовать частоте системного таймера). Этот параметр используется для проверки параметра «тики».

Директива STLO$ —остановить задачу до установки дизъюнкции флагов — останавливает выполнение задачи, издавшей директиву, до тех пор, пока не будет установлен один или несколько указанных флагов из следующих групп:

группа 0 — локальные флаги с 1 по 16;
группа 1 — локальные флаги с 17 по 32;
группа 2 — общие флаги с 33 по 48;
группа 3 — общие флаги с 49 по 64;
группа 4 — групповые глобальные флаги с 65 по 80;
группа 5 — групповые глобальные флаги с 81 по 96.

Если в момент издания директивы какие-либо флаги из числа указанных уже установлены, то выполнение задачи не будет приостановлено.

Директива STLO$ не может быть издана из AST – подпрограммы.

Задача, выполнение которой приостановлено директивой STLO$, может быть выведена из этого состояния только путем установки одного или нескольких флагов; вывести из останова такую задачу директивой USTP$ (вывести задачу из останова), командой MCR UNSTOP или командой DCL START нельзя. Вызов на языке Фортран:

```
CALL STLOR (EF1, EF2, ..., EFN)
CALL STLORS (IDS, EF1, EF2, ..., EFN)
```

Макровызов:

```
STLO$  GRP, MSK
```

Директива STOP$S — остановить выполнение задачи — останавливает задачу, издавшую директиву. Директива STOP$S не может быть издана задачей, находящейся в процессе обслуживания AST – прерывания. Задача, остановленная по STOP$S, может быть выведена из останова только следующим образом:

1) по директиве USTP$ (вывести задачу из останова), изданной другой задачей;
2) по директиве USTP$, изданной этой же задачей из AST – программы;
3) по команде MCR UNSTOP или DCL START

Рекомендуется $S – форма макровызова. Вызов на языке Фортран:

```
CALL STOP ([IDS])
```

Макровызов:

```
STOP$S
```

Директива STSE$ — остановить задачу до установки единичного флага — останавливает задачу, выдавшую директиву, до тех пор, пока не будет установлен указанный флаг. Если в момент выдачи директивы флаг уже установлен, то задача остановлена не будет. Данная директива не может быть издана задачей в процессе обработки AST – прерывания.

Задача, остановленная по директиве STSE$, так же как и в случае директивы STLO$, может быть выведена из останова только путем установки соответствующего флага; директива USTP$, или команда MCR UNSTOP, или команда DCL START не могут вывести такую задачу из останова. Вызов на языке Фортран:

```
CALL STOPFR (EFN [,IDS])
```

Макровызов:

```
STSE$  EFN
```

Директива ULGF$ *—разблокировать групповые глобальные флаги* — уменьшает на единицу счетчик использования групповых глобальных флагов для группы, указанной в UIC по защите задачи (поле H.CUIC+1). Директива разблокирует групповые флаги, которые блокируются при создании по директиве CRGF$ (создать групповые глобальные флаги).

Задача должна разблокировать флаги, прежде чем блокировать их снова.

Групповые глобальные флаги уничтожаются, если выполняются следующие два условия:

счетчик использования в блоке GFB в результате выполнения директивы ULGE$ стал равен нулю;

блок GFB отмечен на удаление.

Рекомендуется $S – форма директивы.

Вызов на языке Фортран:

```
CALL ULGF ([IDS])
```

Макровызов:

```
ULGF$S  [ERR]
```

Директива USTP$ *— вывести задачу из останова* — выводит из останова задачу, выполнение которой было приостановлено директивой STOPS или RCST$. Директива USTP$ неприменима к задаче, ожидающей установки одного флага или дизъюнкции флагов, либо к установленному на время буферизованному вводу–выводу. Если задача была остановлена по директиве STOP$ или RCST$, то в момент издания директивы USTP$ находилась в AST – состоянии, останов задачи будет снят только после завершения обработки AST – прерывания. Директива USTP$ не вызывает объявления важного события. Вызов на языке Фортран:

```
CALL USTP (TSK [,IDS])
```

Макровызов:

```
USTP$  [TSK]
```

Директива WSIG$*S — ждать важного события* — используется для того, чтобы приостановить выполнение задачи, издавшей директиву, до тех пор, пока в системе не произойдет важное событие. Это средство особенно эффективно для приостанова выполнения задачи, которая не может продолжаться из–за недостатка динамической памяти, так как важные события, имеющие место в системе, часто приводят к освобождению динамической памяти. Выполнение директивы само по себе не порождает важного события. Рекомендуется $S – форма директивы. Вызов на языке Фортран:

```
CALL WFSNE
```

Макровызов:

```
WSIG$S  [ERR]
```

Директива WTLO$ *—ждать дизъюнкцию флагов* — приостанавливает выполнение задачи, издавшей директиву, до тех пор, пока не будет установлен один флаг или комбинация любых из указанных флагов событий в одной из групп флагов.

Если при издании директивы какой–либо из указанных флагов установлен, то выполнение задачи не приостанавливается. Вызов на языке Фортран:

```
CALL WFLOR (EF1, EF2, ... EFN)
CALL WFLORS (IDS, EF1, EF2, ..., EFN)
```

Макровызов:

```
WTLO$  GRP, MSK
```

Директива WTSE$ *— ждать единичный флаг* — приостанавливает выполнение задачи, издавшей директиву, до тех пор, пока не будет установлен указанный флаг события. Если при издании директивы этот флаг события уже установлен, выполнение задачи не приостанавливается. Вызов на языке Фортран:

```
CALL WAITER (EFN [,IDS])
```

Макровызов:

WTSE$ EFN

2.3.5. ДИРЕКТИВЫ УПРАВЛЕНИЯ ПРЕРЫВАНИЯМИ

Директивы управления прерываниями обеспечивают возможность обработки программных прерываний для передачи управления в выполняющуюся задачу.

Директива ASTX$S — выход из программы обслуживания AST — обеспечивает выход из обслуживающей AST – программы.

Если в очереди AST – прерываний есть еще AST – прерывания и обслуживание прерываний не запрещено, то сразу после выхода из AST – программы обслуживается следующее AST – прерывание. Если очередь AST – прерываний пуста, то восстанавливается состояние задачи, бывшее до AST–прерывания. Рекомендуется $S – форма директивы. Вызов на языке Фортран отсутствует.

Макровызов:

ASTX$S [ERR]

Директива DSAR$S или IHAR$S —запретить распознавание AST — запрещает распознавание AST–прерываний для задачи, издавшей директиву. AST – прерывания, которые возникают после запрещения распознавания AST – прерываний, ставятся в очередь и будут обработаны, когда распознавание AST–прерываний станет возможным. Предполагается, что во время работы подпрограммы обработки AST–прерываний действует директива запрещения распознавания AST – прерываний. По умолчанию предполагается, что при начальном запуске задачи распознавание AST – прерываний разрешено. Рекомендуется $S – форма директивы. Вызов на языке Фортран:

CALL DSASTR [(IDS)]

или

CALL INASTR [(IDS)]

где IDS — слово для кода завершения директивы.

Макровызов:

DSAR$S [ERR]

или

IHAR$S [ERR]

Директива SCAA$ —назначить программу обработки AST-прерываний по получению командной строки — разрешает или запрещает задаче CLI, издавшей директиву, выполнять обработку AST–прерываний по получению командной строки. Если обработка AST–прерываний по получению командной строки разрешена, то при передаче командной строки задаче CLI управляющая программа передает управление на соответствующую AST–программу. Директива SCAA$ может быть использована только задачами CLI.

При входе в AST–программу стек задачи содержит следующую информацию:

SP + 10 — ноль, так как флаги не используются;
SP + 06 — PS задачи в точке AST-прерывания;
SP + 04 — PC задачи в точке AST-прерывания;
SP + 02 — DSW в момент AST-прерывания;
SP + 00 — адрес только что переданного задаче командного буфера.

Перед выходом на директиву ASTX$S (выход из программы обслуживания AST) AST–программа должна удалить из стека адрес командного буфера.

Адресом командного буфера является адрес буфера, указанный в директиве GCCI$ (дать команду для CLI). Вызов на языке Фортран отсутствует.

Макровызов:

SCAA$ [AST]

Директива SFPA $ — назначить программу обработки AST-прерываний по ошибкам команд с плавающей запятой — разрешает обработку AST–прерываний по ошибкам команд с плавающей

запятой и задает адрес программы обслуживания AST–прерывания этого типа или извещает систему, что AST–прерывания по ошибкам команд с плавающей запятой для задачи, издавшей директиву, игнорируются. Вызов на языке Фортран отсутствует.

Макровызов:

```
SFPA$  [AST]
```

Директива SPRA$ —AST-прерывание по восстановлению питания — разрешает обработку AST-прерываний по восстановлению питания и задает адрес точки входа программы, обслуживающей AST-прерывание этого типа или извещает систему, что для задачи, издавшей директиву, AST-прерывание по восстановлению питания игнорируется.

Если адрес точки входа в обслуживающую AST-программу задан, то AST-прерывания по восстановлению питания для задачи, издавшей директиву, допустимы, и, если такое AST-прерывание произойдет, управление будет передано на программу обработки. Вызов на языке Фортран отсутствует.

Макровызов:

```
SPRA$  [AST]
```

Директива SRDA$ —AST-прерывание по получению данных— разрешает обработку AST-прерывания по получению данных и задает адрес точки входа в программу обслуживания AST-прерывания этого типа или указывает системе, что AST-прерывания по получению данных для задачи, издавшей директиву, больше не обрабатываются. Вызов на языке Фортран отсутствует.

Макровызов:

```
SRDA$  [AST]
```

Директивы SREA$ и SREX$ —AST- прерывание по аварийному завершению задачи — разрешают обработку AST-прерываний, объявляемых по аварийному завершению задачи, и задают адрес AST-программы, обслуживающей AST-прерывание этого типа. Аварийное завершение задачи может произойти по директиве ABRT$ или по команде ABORT MCR или DCL.

Директивы SREA$ и SREX$ позволяют выполнить определенные завершающие действия при аварийном завершении задачи.

Если адрес AST-программы в директивах не задан, то ранее заданный адрес AST-программы удаляется. Вызов на языке Фортран:

```
CALL AREA (AST [,IDS])
```

Макровызов:

```
SREA$  AST
SREX$  [AST]  [,RESERV]
```

Директива SRRA$ —AST-прерывание по получению данных по ссылке — выполняет одну из двух функций:

1) разрешает AST-прерывания по получению данных, передаваемых по ссылке, для задачи, издавшей директиву. Она задает адрес AST-программы, на которую управляющая программа передает управление каждый раз, когда происходит AST-прерывание этого типа;

2) запрещает AST-прерывание по получению данных по ссылке для задачи, издавшей директиву; адрес программы обработки AST-прерывания «обнуляется». Вызов на языке Фортран отсутствует.

Макровызов:

```
SRRA$  [AST]
```

Директива SVDB$ —задать таблицу SST-векторов для отладочных средств — задает адрес таблицы точек входов программ, обслуживающих SST-прерывания для отладочных средств, используемых задачей (например, для отладчика). При отказе от обработки SST-прерываний отладчиком издается макровызов без параметров.

Если точка входа программы обслуживания SST-прерывания определена как в таблице SST-векторов, заданной для задачи, так и в таблице SST-векторов, заданной для отладочных средств, но не

для задачи. Вызов на языке Фортран отсутствует.

Макровызов:

```
SVDB$  [ADR]  [,LEN]
```

Директива SVTK$ —задать таблицу SST-векторов для задачи — задает адрес таблицы точек входов программ, обслуживающих SST-прерывания для задачи, издавшей директиву.

Макровызов без параметров означает отказ от обработки SST-прерывания. Вызов на языке Фортран отсутствует.

Макровызов:

```
SVTK$  [ADR]  [,LEN]
```

2.3.6. ДИРЕКТИВЫ ВВОДА-ВЫВОДА И МЕЖЗАДАЧНЫХ СВЯЗЕЙ

Директивы этой группы обеспечивают задаче доступ к устройствам ввода-вывода на уровне драйверного интерфейса или на уровне прерываний, связь с другими задачами в системе, прием командной строки, посланной задаче через MCR или DCL при запуске задачи.

Директива ALUN$—назначить логический номер устройства — назначает LUN физическому устройству. Это означает, что задача не присоединяет данное физическое устройство, а использует данный логический номер для обращения к физическому устройству. Вызов на языке Фортран:

```
CALL ASNLUN (LUN, DEV, UNT [,IDS])
```

Макровызов:

```
ALUN$  LUN, DEV, UNT
```

Директива CINT$ —подключиться к вектору прерывания — позволяет задаче самой обрабатывать прерывания от устройства по указанному вектору. Программа обслуживания прерываний (ISR) размещается в теле задачи. В системе с диспетчером памяти эту директиву может издавать только привилегированная задача.

Прерывание обслуживается на трех уровнях: прерывания устройства, отложенного прерывания и на уровне задачи. На последнем уровне различаются два подуровня: AST-прерывание и не AST-прерывание. Вызов на языке Фортран отсутствует.

Макровызов:

```
CINT$  VEC, BASE, ISR, EDIR, PRI, AST
```

Для отключения от вектора прерываний аргумент ISR должен быть равен нулю; аргументы BASE, EDIR и AST в этом случае игнорируются.

Директива GLUN$—дать информацию об устройстве с заданным логическим номером — заполняет 6-словный буфер информацией о физическом устройстве, логический номер которого указан в директиве. Если физическое устройство переназначено, то возвращаемая информация будет описывать фактически назначенное устройство. Вызов на языке Фортран:

```
CALL GETLUN (LUN, DAT [,IDS])
```

где DAT — 6-словный целый массив для получения информации об устройстве.

Макровызов:

```
GLUN$  LUN, BUF
```

где BUF—адрес 6-словного буфера, в котором возвращается информация об устройстве.

Директива GMCR$ —дать командную строку — позволяет системе передать 80-байтную командную строку задаче, издавшей директиву.

Если задача установлена с именем «...имя» или «имя TNN» (имя состоит из трех алфавитно-цифровых символов, NN — восьмеричный номер терминала, с которого подана команда), то MCR будет запрашивать выполнение этой задачи при вводе с терминала NN команды в виде

```
>имя командная строка
```

Задача, вызванная таким способом, должна издать директиву GMCR, которая передаст командную строку в 80-байтный буфер задачи. Вызов на языке Фортран:

```
CALL GETMCR (BUF[, IDS])
```

где BUF—80-байтный массив для получения командной строки.

Макровызов:

GMCR$

$S-форма директивы GMCR$ не поддерживается, так как командная строка принимается в блок DPB.

Система преобразует командную строку следующим образом:

- заменяет табуляцию на единичный пробел;
- заменяет несколько пробелов одним;
- преобразует строчные латинские буквы в латинские прописные;
- удаляет все пробелы в конце строки;
- удаляет комментарии.

Последним символом в строке является ограничитель строки <CR> или <ESC>.

Директива QIO$ —поставить в очередь запрос ввода-вывода— ставит запрос ввода-вывода в очередь запросов, упорядоченную по приоритетам, к указанному физическому устройству. Физическое устройство задается логическим номером (LUN).

При завершении операции ввода-вывода управляющая программа объявляет важное событие. Если в директиве указан флаг события, то при включении запроса в очередь он «обнуляется». При завершении обработки этого запроса флаг устанавливается. Когда запрос ставится в очередь, очищается блок состояния ввода-вывода, а после выполнения этого запроса в него засылается код завершения ввода-вывода.

Если адрес AST-программ задан, то по завершении ввода-вывода будет объявлено AST-прерывание; программа обработки AST-прерывания получит в стеке задачи, издавшей директиву, слово маски ожидаемых флагов событий, PS, PC задачи в точке AST-прерывания, слово состояния директивы и адрес блока состояния ввода-вывода. Вызов на языке Фортран:

CALL QIO (FNC, LUN, [EFN], [PRI], [ISB], [PRL], [,IDS])

Макровызов:

QIO$ FNC, LUN, [EFN], [PRI], [ISB], [AST], [PRL]

Директива QIOW$ —поставить в очередь запрос ввода-вывода и ждать его завершения — подобна директиве QIO$ и отличается от нее только одним: если в директиве QIOW$ указан флаг события, то к директиве QIQ$ автоматически добавляется директива WTSE$ (ждать единичный флаг). Если флаг не задан, то директива QIOW$ выполняется так же, как и директива QIO$. Вызов на языке Фортран:

CALL WTQIO (FNC, LUN, EFN, [PRI], [ISB], [PRL] [,IDS])

Макровызов:

QIOW$ FNC, LUN, [EFN], [PRI], [ISB], [AST] [,PRL]

Директива RCST$ —получить данные или остановить задачу — позволяет получить из очереди данных задачи, издавшей директиву, 13-словный блок данных. Блок данных был помещен в очередь задачи по директиве SDAT$ (послать данные) или по директиве SDAT$ (послать данные, запросить выполнение, подключиться). Вызов на языке Фортран:

CALL RCST ([TSK], IBUF[, IDS])

где BUF — имя 15-словного массива, в котором задача получит имя задачи, пославшей данные, и сами данные.

Макровызов:

RCST$ [TSK], BUF

где BUF—адрес 15-словного буфера, в котором задача получит имя задачи, пославшей данные, и сами данные.

Директива RCVD$ —получить данные — позволяет получить из очереди данных задачи, издавшей директиву, 13-словный блок данных, который был поставлен в очередь другой задачей с помощью директивы SDAT$ (послать данные). Порядок выполнения очереди — «первый пришел, первый обслужен». Директива RCVD$ передает задаче в указанном 15-словном буфере имя посылающей

задачи (2 слова в RADIX-50) и 13 слов данных. Имя посылающей задачи содержится в первых двух словах блока.

Если директиву RCVD$ издает подчиненная задача, она получает UIC (если только у нее нет активных групповых глобальных флагов) и TI: задачи, издавшей директиву SDAT$. Вызов на языке Фортран:

```
CALL RECEIV ([TSK], BUF [„IDS])
```

где BUF— 15-словный целый массив для получения данных.

Макровызов:

```
RCVD$   [TSK], BUF
```

где BUF—адрес 15-словного буфера для получения данных.

Директива RCVX$ —получить данные или выйти — позволяет получить из очереди данных задачи, издавшей директиву, 13-словный блок данных, который был поставлен в эту очередь другой задачей с помощью директивы SDAT$ (послать данные), либо завершает задачу, если данные не были посланы. Дисциплина очереди — «первый пришел, первый обслужен».

Имя посылающей задачи (два слова в RADIX-50) и 13-словный блок данных передаются в 15-словный буфер, указанный в директиве; имя посылающей задачи занимает первые два слова буфера.

В связи с тем, что возможно завершение задачи (если очередь пуста), необходимо позаботиться о завершении запросов ввода-вывода и закрытии файлов.

Если подчиненная задача издает директиву RCVX$, она получает UIC (если только у нее нет активных групповых глобальных флагов) и TI: задачи, пославшей данные. Вызов на языке Фортран:

```
CALL RECOEX ([TSK],BUF [„IDS])
```

где BUF— 15-словный целый массив для получения данных.

Макровызов:

```
RCVX$   [TSK], BUF
```

где BUF—адрес 15-словного буфера для получения данных.

Директива SDAT$ —послать данные — позволяет поставить в очередь получающей задачи 13-словный блок данных (порядок выполнения очереди — «первый пришел, первый обслужен») и объявляет важное событие. Если указан локальный флаг события, то он устанавливается для посылающей задачи. Вызов на языке Фортран:

```
CALL SEND (TSK, BUF,  [EFN]   [,IDS])
```

где BUF— 13-словный целый массив посылаемых данных.

Макровызов:

```
SDAT$   TSK, BUF [,EFN]
```

где BUF—адрес 13-словного блока данных.

2.3.7. ДИРЕКТИВЫ УПРАВЛЕНИЯ ПАМЯТЬЮ

Директивы управления памятью позволяют задаче манипулировать виртуальным и логическим адресным пространством, динамически создавать и контролировать состояние районов и адресных окон. Директивы этой группы позволяют задачам обмениваться данными, передавая друг другу информацию о размещении данных в районе.

Директива ATRG$ —присоединить район — присоединяет к задаче, издавшей директиву, статический общий район либо именованный динамический район (с помощью этой директивы другие районы не могут быть присоединены к задаче). Управляющая программа проверяет совместимость запрошенного доступа, заданного в слове состояния района, с правами на доступ к району, определенными для той категории пользователей, к какой относится задача. Если запрошенный тип доступа разрешен, то район присоединяется к задаче, и управляющая программа возвращает в поле R.GID блока RDB 16-битный ID района. ID района используется задачей в последующих директивах управления памятью. Если район присоединен к задаче, директива ATRG$ используется только для получения ID района. Вызов на языке Фортран:

CALL ATRG (RDB [,IDS])

Макровызов:

ATRG$ RDB

Директива CRAWS — создать адресное окно — создает новое виртуальное адресное окно и заполняет блок окна в заголовке задачи, создавая виртуальный базовый адрес и размер окна. Место для блока окна в заголовке задачи резервируется при построении задачи по параметру WNDWS. Если новое окно перекрывает виртуальные адреса ранее созданных окон, то для старых окон прекращается отображение и они уничтожаются. При успешном создании окна задаче возвращается 8-битный идентификатор окна (ID окна).

ID окна, возвращаемый задаче, — это число от 1 до 23; ID окна является индексом блока окна в заголовке задачи. Блок окна описывает созданное адресное окно. Вызов на языке Фортран:

CALL CRAW (WDB [,IDS])

Макровызов:

CRAWS WDB

Директива CRRG$ —создать район — создает динамический район в системно-управляемом разделе и, в зависимости от параметров, может присоединить его к задаче, издавшей директиву.

Управляющая программа возвращает код ошибки, если для размещения района в указанном разделе недостаточно места. Вызов на языке Фортран:

CALL CRRG (RDB [,IDS])

Макровызов:

CRRG$ RDB

Директива DTRG$ —отсоединить район — отсоединяет ранее присоединенный район от задачи, издавшей директиву. При этом автоматически прекращается отображение всех окон, отображенных ранее в район. Вызов на языке Фортран:

CALL DTRG (RDB [,IDS])

Макровызов:

DTRG$ RDB

Директива ELAW$ —уничтожить адресное окно — уничтожает указанное в директиве адресное окно. Если в момент уничтожения производится отображение адресного окна, то отображение прекращается. Последующее использование ID уничтоженного окна недопустимо. Вызов на языке Фортран:

CALL ELAW (WDB [, IDS])

Макровызов.

ELAW$ WDB

Директива GMCX$ —дать описание окон — возвращает описание существующих в текущий момент адресных окон. Описание возвращается в формате, который дает возможность пользователю восстановить отображение адресных окон с помощью серии директив CRAW$. Аргументом GMCX$ является адрес таблицы, содержащей незаполненные для создания блоки WDB, и слово-ограничитель таблицы (для каждого блока окна, размещенного в заголовке задачи, должен быть задан свой блок WDB). Вызов на языке Фортран:

CALL GMCX (IMCX [,IDS])

где IMCX — целый массив для получения описания существующих в текущий момент адресных окон.

Макровызов:

GMCX$ WVEC

Директива MAP$ — отобразить адресное окно — отображает адресное окно в присоединенный район. Отображение начинается с заданного в директиве места района. Если окно уже отображается, то,

прежде чем выполнить требуемое отображение, управляющая программа прекращает действующее отображение.

Длина, задаваемая для отображения, должна быть меньше или равна минимальной из следующих величин:

размеру окна, указанному при создании окна;

длине, оставшейся между указанным в директиве началом отображения в районе и концом района.

Чтобы иметь возможность отображать с правом доступа «на запись», район должен быть присоединен с правом доступа на запись; чтобы можно было выполнять отображения с правом доступа «только чтение», район должен быть присоединен с правом доступа на чтение либо на запись. Вызов на языке Фортран:

```
CALL MAP (WDB  [,IDS])
```

Макровызов:

```
MAP$  WDB
```

Директива RREF$—получить по ссылке — выбирает и передает задаче элемент из очереди получения по ссылке получившей задачи. В зависимости от заданных параметров задача может завершиться, если очередь получений по ссылке окажется пустой. В директиве может быть также указано, что управляющая программа должна выполнить отображение виртуального адресного окна в область, в которой передаются данные. Если директива завершена успешно, объявляется важное событие.

Каждый элемент очереди, полученный по ссылке, требует отдельного присоединения района к соответствующей задаче.

Район, в котором передано несколько посылок для задачи, присоединяется к ней столько же раз. Присоединение района требует системной динамической памяти: для ее экономии получающая посылку задача должна отсоединить ранее присоединенный ею район, т. е. каждый район присоединяется к задаче только один раз. Вызов на языке Фортран:

```
CALL RREF (WDB, [BUF]   [,IDS])
```

где BUF—10-словный целый массив, используемый как дополнительный буфер данных.

Макровызов:

```
RREF$  WDB
```

Директива RRST$ — получить по ссылке или остановиться — выбирает и передает задаче элемент из очереди получения по ссылке получающей задачи. Если очередь пуста, то задача будет остановлена. В директиве может быть также указано, что управляющая программа должна выполнить отображение виртуального адресного окна в область, в которой передаются данные.

Если директива завершена успешно, то объявляется важное событие. Вызов на языке Фортран:

```
CALL RRST (WDB, [BUF]   [,IDS])
```

где BUF— 10-словный целый массив, используемый как дополнительный буфер данных.

Макровызов:

```
RRST$  WDB
```

Директива SREF$ — послать данные посредством передачи ссылки — ставит элемент, содержащий информацию, необходимую для обращения к району, в очередь получений по ссылке указанной (получающей) задачи. По каждой директиве SREF$, изданной задачей, управляющая программа автоматически присоединяет район, на который делается ссылка, к получающей задаче. Вызов на языке Фортран:

```
CALL SREF (TSK,  [EFN], WDB, [BUF]   [,IDS])
```

где BUF—8-словный целый массив, содержащий дополнительную информацию. Если такой массив задан, то его адрес содержится в восьмом слове блока WDB. Если этот адрес опущен, то содержимое восьмого слова блока WDB не изменяется.

Макровызов:

```
SREF$    TSK, WDB  [,EFN]
```

Директива UM AP$ *— прекратить отображение адресного окна* — прекращает отображение заданного адресного окна. После того как отображение окна прекращено, обращение к виртуальным адресам окна недопустимо и вызывает прерывание процессора. Вызов на языке Фортран:

CALL UNMAP (WDB [,IDS])

Макровызов:

UMAP$ WDB

2.3.8. ДИРЕКТИВЫ ПОРОЖДЕНИЯ ЗАДАЧ

Для синхронизации выполнения задач реального времени большую роль играет механизм, устанавливающий связь задач друг с другом. Он основан на понятиях «задача-родитель» и «задача-потомок».

Задача-родитель — это задача, которая либо «порождает» другую задачу, либо «подключается» к другой задаче, называемой *задачей-потомком.*

Порождение задачи означает активизацию одной задачи из другой и установление связи между этими задачами, называемыми задачей-родителем и задачей-потомком.

Подключение к задаче представляет собой установление связи между двумя активными задачами — задачей-родителем и задачей-потомком.

Связь «родитель — потомок» подразумевает, что задача-родитель может быть оповещена о завершении задачи-потомка; кроме того, задача-родитель получает код завершения задачи-потомка либо информацию о текущем состоянии задачи-потомка. Возвращаемый код состояния сообщает об успешном завершении задачи-потомка либо указывает на возникновение ошибочной ситуации при ее выполнении.

Отношения «родитель — потомок» полезны для организации пакетной обработки.

Директивы межзадачных отношений «родитель — потомок».

Директивы, порождающие между задачами отношения «родитель — потомок», делятся на две группы:

директивы, создающие эти отношения;

директивы, передающие отношения «родитель — потомок» другой задаче.

Для создания отношений «родитель — потомок» используются директивы SPWN$, CNCT$, SDRC$.

К директивам, передающим отношения «родитель — потомок» другим задачам, относятся директивы RPOI$ и SDRP$.

Директивы связи задач. Задача-потомок может вернуть задаче-родителю информацию о своем состоянии с помощью директив EXST$ и EMST$

Директива CNCT$ *— подключиться к задаче* — позволяет синхронизировать выполнение задачи, издавшей директиву, с завершением или с состоянием другой задачи (потомка), активной в момент издания директивы CNCT$. Она устанавливает связь с другой задачей (задачей-потомком) следующим образом: создает блок ОСВ и ставит его в очередь к задаче-потомку. Кроме того, директива увеличивает на единицу счетчик задач-потомков в ТСВ задачи (поле T.RDCT), издавшей директиву. Счетчик задач-потомков указывает число задач, к которым подключена данная задача, и общее количество виртуальных терминалов, созданных данной задачей. При завершении задачи-потомка или при выдаче ею состояния для задачи-родителя издается AST-прерывание; соответствующая AST-программа получает на входе в стеке задачи адрес блока состояния, содержащего код состояния задачи-потомка.

Данная директива не должна издаваться для подключения к задаче, являющейся интерпретатором командных строк. Вызов на языке Фортран:

CALL CNCT (TSK, [EFN], [AST]. [ESB]. [PARM] [.IDS])

CALL CNCTN (TSK, [EFN], [AST], [ESB], [PARM] [IDS])

где PARM — имя слова, в котором передается адрес блока состояния задачи-потомка для AST-подпрограммы.

Макровызов:

CNCT$ TSK, EFN ,[EAST], [ESB]

Директива EMST$ *—выдать состояние* — передает указанной в директиве задаче-родителю

информацию о состоянии задачи в виде 16-битного слова. Директива устанавливает также флаг и/или объявляет AST-прерывание, если при подключении к данной задаче по директивам SDRC$ (послать данные, запросить выполнение, подключиться), SPWN$ (породить задачу) и CNCT$ (подключиться к задаче) были заданы флаг и (или) AST-программа.

Если указанная в директиве EMST$ задача многократно подключалась к задаче, издавшей директиву, то для возвращения состояния задаче-родителю используется первый (самый старый) из соответствующих блоков ОСВ. Если в директиве опущено имя задачи, то эти действия выполняются для всех задач, подключенных в данный момент к задаче, издавшей директиву. После передачи информации о состоянии задачи задаче-родителю (родителям) директива уничтожает подключение. Вызов на языке Фортран:

CALL EMST ([TSK], STATUS [,IDS])

Макровызов:

EMST$ [TSK], STATUS

Директива EXST$ —завершить задачу и выдать информацию о состоянии — завершает выполнение задачи, издавшей эту директиву, и передает 16-битный код завершения задачи (сосостояние задачи) всем задачам, подключенным к ней по директивам SPWN$, CNCT$ или SDRC$. Если подключенных задач нет, то EXST$ завершает выполнение задачи. Управляющая программа не накладывает никаких ограничений на формат кода состояния; этот формат известен только задаче-родителю и задаче-потомку. Однако если задача-потомок завершается аварийно, то задаче-родителю возвращается код EX$SEV. Это значение интерпретируется процессорами пакетной обработки как «грубая ошибка». В тех случаях, когда задача издает обычную директиву завершения, всем задачам, подключенным к ней, передается код EX$SUC — код успешного завершения. Вызов на языке Фортран:

CALL EXST (STATUS)

Макровызов:

EXST$ STATUS

Директива RPOI$ —запросить управление и передать информацию о подключении — запрашивает на выполнение указанную задачу и подключает к ней одного или всех «родителей» задачи, издавшей директиву. Директива позволяет передать командную строку в данную задачу. Только привилегированные задачи и интерпретатор CLI могут указывать UIC и TI: запрашиваемой задачи. Вызов на языке Фортран:

CALL PROI$ (TSK, [UGC], [UMC], [PARENT], [BUF], [BFL], [SC], [DEV], [UNT), [TASK], [OCBAD], [IDS])

Макровызов:

PROI$ TSK ... [UGC], [UMC], [PARENT], [BUF], [BFE], [SC], DEV], [UNT], [TASK], [OCBAD]

где BUF—буфер, передаваемый запрашиваемой задаче;
BFE — длина буфера, передаваемого запрашиваемой задаче в байтах.

Директива SDRC$ — послать данные, запросить выполнение, подключиться — выполняет следующие функции: посылает данные указанной задаче, запускает ее на выполнение, если она еще неактивна, и подключается к ней. Получающая задача обычно возвращает код своего состояния через директиву EMST$ (выдать состояние) или EXST$ (завершить задачу и выдать состояние). Вызов на языке Фортран:

CALL SDRC (TSK, BUF, [EFN], [AST], [ESB], [PARM] [,IDS]) CALL SDRCN (TSK, BUF, [EFN], [AST], [ESB], [PARM] [,IDS])

где BUF—имя 13-словного посылаемого буфера;
PARM — имя слова, в котором передается адрес блока состояния для AST-подпрограммы.

Макровызов:

SDRC$ TNAME, BUF, [EFN], [EAST], [ESB]

где BUF—адрес 13-словного буфера данных, посылаемых запрашиваемой задаче.

Директива SDRP$ —послать данные запросить выполнение, передать ОСВ — посылает указанной

задаче данные, запрашивает эту задачу на выполнение, если она еще неактивна, и передает ей своих родителей. Вызов на языке Фортран:

```
CALL SDRP (TSK, BUF, [BFL], [EFN], [IFLAG], [PARENT], [OCBAD]  [,IDS])
```

где BUF — имя целого массива, содержащего посылаемые данные;
BFL — имя целой переменной, содержащей количество слов, посылаемых запрашиваемой задаче. Аргумент может принимать значения от 1 до 255. По умолчанию принимается значение, равное 13 (10);
IFLAG — имя целой переменной, содержащей флаговые биты, управляющие выполнением директивы.

Макровызов:

```
SDPR$  TSK, BUF,  [BFE],  [EFN],  [FLAG],  [PARENT],  [OCBAD]
```

где BUF — адрес буфера, содержащего посылаемые данные;
BFE — длина буфера;
FLAG — флаговые биты, управляющие выполнением директивы.

Директива SPWN$ —породить задачу — выполняет следующие функции: издает запрос на выполнение указанной в директиве задачи, ставит в очередь командных строк переданную в директиве строку, назначает в качестве TI: для активизируемой задачи либо предварительно созданный виртуальный терминал, либо физический терминал.

Директива SPWN$ порождает указанную задачу: создает блок OCB, ставит его в очередь блоков OCBK указанной (порожденной) задаче, увеличивает счетчик порожденных задач в блоке TCB задачи, издавшей директиву.

Счетчик порожденных задач указывает управляющей программе, что задача является задачей-родителем и имеет одного или более потомков и виртуальных терминалов. Очистка счетчика необходима, если при завершении задачи-родителя остаются активные задачи-потомки. При завершении задачи-потомка счетчик порожденных задач уменьшается на единицу.

Блок OCB содержит адрес TCB задачи-родителя, а также всю информацию, необходимую для обработки события, связанного с завершением порожденной задачи.

Если в директиве задана командная строка, то она сохраняется в системной динамической области и ставится в системную очередь командных строк. Порожденная задача получает командную строку по директиве GMCR$ (получить командную строку). Максимальная длина командной строки равна 255 (10) байтам.

Если в директиве задан адрес AST-программы, то по завершении порожденной задачи для задачи-родителя объявляется AST-прерывание. На входе AST-программа получает в стеке задачи адрес блока состояния, содержащего код завершения порожденной задачи. Перед входом в директиву ASTX$ (выход из программы обслуживания AST) AST-программа должна удалить из стекла адрес блока состояния задачи.

Если порождаемая задача является CLI, то директива SPWN$ выполняет следующие действия: командная строка (она должна быть обязательно указана в директиве) и блок OCB ставятся в очередь к интерпретатору CLI. Он либо сам обрабатывает командную строку, либо передает ее другой задаче. В том случае, когда командная строка обрабатывается непосредственно самим интерпретатором CLI, блок OCB используется, чтобы имитировать немедленное завершение интерпретации и передать породившей задаче через соответствующую управляющую подпрограмму код ее завершения. Если интерпретатор CLI передает командную строку другой задаче, то он просто переставляет блок OCB из своей очереди в очередь блока OCB к той задаче, которой предназначена командная строка. Командная строка переставляется в очередь командных строк задачи таким образом, как если бы она была послана задаче непосредственно. Таким образом, задача, фактически получающая командную строку, находится в таком же состоянии, как если бы она была явно указана в директиве SPWN$. Условия завершения порожденной задачи не возникают до тех пор, пока задача, фактически получившая командную строку, не завершится. Вызов на языке Фортран:

```
CALL SPAWN  (TSK,  [UGC],  [UMC],  [EFN],  [AST],  [ESB],  [PARM], [ICMLIN, ICMLEN], [UN]  [DEV]  [,IDS]
CALL SPAWNN (TSK, [UGC],  [UMC], [EFN],'[AST],  [ESB],  [PARM],[ICMLIN, ICMLEN],  [UN],  [DEV],  [,IDS]
```

где PARM — имя слова, в котором для AST-программы передается адрес блока состояния порожденной задачи;
ICMLIN — имя массива, содержащего командную строку для порождаемой задачи;
ICMLEN — длина командной строки (максимальная длина — 255 (10) байт).

Макровызов:

SPWN$ TSK ,,, [UGC], [UMC], [EFN], [EAST], [ESB], [CMDLIN, CMDLEN], [UNUM], [DNAM]

где CMDLIN — адрес командной строки для порождаемой задачи;
CMDLE — длина командной строки (максимальная длина 255 (10) байт).

2.3.9. ДИРЕКТИВЫ, ПРЕДНАЗНАЧЕННЫЕ ДЛЯ ЗАДАЧИ CLI

Данная группа директив позволяет задачам CLI получать командные строки, запрашивать и передавать информацию о подключении, получать информацию о задачах CLI, назначать их терминалу.

Директива GCCI$ —дать команду для CLI — переписывает командную строку CLI из системного буфера в буфер задачи. Задаче также передается информация о терминале, с которого получена команда.

Директиву GCCI$ могут использовать только задачи, являющиеся интерпретаторами командных строк. Вызов на языке Фортран:

CALL GTCMC1 (ICBF, ICBFL, [IIBUF], [IIBFL], [IADDR], [INCP] [,IDS])

где ICBF — имя командного буфера, куда передается командная строка;
ICBFL — имя целой переменной, содержащей длину командного буфера в байтах;
IIBUF — имя целого массива (информационного буфера), куда передается вспомогательная информация для задачи CLI;
IIBFL — имя целой переменной, содержащей адрес системного буфера, в котором находится команда для задачи CLI. Этот адрес оставляет в информационном буфере предыдущий вызов программы GTCMCI, если в слове маски в предыдущем вызове был задан бит GC.CND;
INCP — имя целой переменной, содержащей битовую маску, которая указывает, какие действия предпринять управляющей программе, если очередь командных строк пуста.

Макровызов:

GCCI$ CBUF, CBFL, [IBUF], [IBFL], [ADDR], [NCP]

где CBUF — адрес командного буфера, куда передается командная строка;
CBFL — длина командного буфера. Максимальная длина буфера равна 266 (10) байтам;
IBUF — адрес буфера для получения информации о терминале, с которого получена команда;
IBFL — длина информационного буфера;
ADDR — адрес команды в системном буфере. Этот адрес был возвращен предыдущей директивой GCCI$ в поле G.CCCA информационного буфера, если в слове маски макровызова задан бит GC.CND. Если в вызове директивы параметр ADDR задан, то задаче передается команда, находящаяся в системном буфере по заданному адресу, а затем команда удаляется из системной очереди или остается в ней (GC.CND). Адрес командной строки передается в информационный буфер задачи лишь в том случае, если команда не удаляется из системной очереди;
NCP — битовая маска, определяющая действие управляющей программы, если очередь командных строк пуста.

Директива GCLI$ — дать информацию об интерпретаторе — передает в буфер задачи информацию о CLI, указанном в директиве, либо о CLI, связанном с данным терминалом. Только привилегированная задача может указать CLI, не связанный с TI: задачи, или терминал, не являющийся устройством TI: задачи. Вызов на языке Фортран:

CALL GETCLI (IBUF, IBFL, [ICLI], [DEV], [UNT], [,IDS])

где IBUF — имя целого массива (буфера), куда посылается информация о CLI;
IBFL — длина буфера в байтах;
ICLI — имя 2-словного массива, содержащего имя CLI в коде RADIX-50;

Макровызов:

GCLI$ BUF, BUFL, [CLI], [DEV] [,UNT]

где BUF — адрес буфера, куда посылается информация о CLI;
BUFL — длина буфера;
CLI — имя CLI в коде RADIX-50.

Директива SCLI$ — назначить CLI терминалу — назначает интерпретатор CLI указанному терминалу.

Директива SCLI$ может быть издана только привилегированной задачей или задачей CLI.

Если в слове состояния CLI указан флаг ограничения доступа (CP.RST), то только сама задача CLI может назначить себя к терминалу. Вызов на языке Фортран:

CALL SETCLI (ICLI, DEV, UNT [,IDS])

где ICLI — имя 2-словного массива, содержащего имя CLI, назначаемого терминалу;
IDS — слово для кода завершения директивы.

Макровызов:

SCLI$ CLI, [DEV], [UNT]

где CLI — имя CLI, назначаемого терминалу.

2.3.10. ДИРЕКТИВЫ ДОПОЛНИТЕЛЬНЫХ ВОЗМОЖНОСТЕЙ

Директивы, использующие дополнительные возможности, позволяют работать с виртуальными терминалами, логическими именами, библиотекой режима супервизора, буферами переменной длины для передачи или получения данных по ссылке.

Директива CLON$ *— создать логическое имя* — устанавливает связь между логическим именем и эквивалентным именем. Максимальная длина каждого параметра — 255(10) символов. Если указывается существующее логическое имя, то устанавливается новое значение логического имени. Вызов на языке Фортран:

CALL CRELON (MOD, TBNUM, LNS, LNSSZ, ENS, ENSSZ [,IDS])

Макровызов:

CLON$ MOD, <PRMLST>, LNS, LNSSZ, ENS, ENSSZ

где <PRMLST> = < [TBNUM] , [STATUS] > —(угловые скобки не требуются, если указан только параметр TBNUM);

STATUS — статус логического имени.

Директива CRVT$ *—создать виртуальный терминал* — создает виртуальный терминал, используемый задачей-родителем для связи с задачей-потомком. Когда задача-потомок осуществляет запись или чтение на свой терминал TI:, запрос посылается задаче-родителю через виртуальный терминал. Например, когда процессор пакетной обработки загружает задачу, он связывается с ней через виртуальный терминал, а не через физический. Вызов на языке Фортран:

CALL CRVT ([IAST], [OAST], [AAST], [MLEN], IPARM [,IDS])

где IPARM — адрес 3-словного буфера для получения информации из стека при появлении AST-прерывания.

Макровызов:

CRVT$ [IAST], [OAST], [AAST], [MLEN]

Директива DLON $ *— удалить логическое имя* — удаляет логическое имя из таблицы логических имен и возвращает системе ресурсы, использованные этим логическим именем. Вызов на языке Фортран:

CALL DELLON (MOD, TBNUM, LNS, LNSSZ [, IDS])

Макровызов:

DLON$ MOD, TBNUM, LNS, LNSSZ

Директива ELVT $ *—удалить виртуальный терминал* — помечает структуры данных виртуального терминала на удаление, отсоединяет виртуальный терминал от списка устройств и удаляет его. Эта директива может быть издана только задачей, создавшей виртуальный терминал. Любая активная непривилегированная задача, TI: которой удаляется, завершается аварийно. Сообщения TKTN, извещающие об аварийном завершении таких задач, направляются на CO:. Все логические номера устройств, назначенные задачей, издавшей директиву, или завершенной задачей-потомком, отменяются. Вызов на языке Фортран:

CALL ELVT (IUNUM [,IDS])

где IUNUM — номер виртуального терминала.

Макровызов:

ELVT$ UNUM

Директива SCAL$ *—вызвать супервизор* — используется задачей в режиме пользователя или супервизора для вызова программы из библиотеки режима супервизора. Возврат в режим пользователя производится после завершения выполнения программы, выполнявшейся в режиме супервизора. Для

этой директивы используется только $S- форма. Вызов на языке Фортран отсутствует.

Макровызов:

SCAL$S SADDR, CADDR [,ERR]

где SADDR — адрес программы из библиотеки режима супервизора;
CADDR — адрес программы завершения для возврата в вызывающую программу.

Директива TLON$ —транслировать логическое имя — возвращает эквивалентную строку, связанную с указанным логическим именем. Вызов на языке Фортран:

CALL TRALON (MOD, TBMSK, LNS, LNSSZ, ENS, ENSSZ, [RSIZE], [RTBMOD], [STATUS], [, IDS])

Макровызов:

TLON$ MOD, TBMSK, LNS, LNSSZ, ENS, ENSSZ, [RSIZE], [RTBMOD], [STATUS]

Директива VRCD$ — получить данные переменной длины – выбирает из очереди задачи, издавшей директиву, блок данных переменной длины и передает его в указанный пользователем буфер. Постановка данных в очередь осуществляется директивой VSDA$ (послать данные переменной длины). Порядок выполнения очереди — «первый пришел, первый обслужен». Вызов на языке Фортран:

CALL VRCD ([TSK], BUFADR, BUFLEN [, IDSW])

где BUFADR — имя массива для получения имени посылающей задачи и данных (должен быть выравнен на границу слова);
BUFLEN — длина буфера.

Макровызов:

VRCD$ [TSK], BUFADR [, BUFLEN]

где BUFADR — адрес буфера;
BUFLEN — размер буфера в словах.

Максимальный размер буфера — 256(10) слов. Если размер буфера не указан (только для макровызова), размер буфера равен 13(10) словам.

Директива VSDA$—послать данные переменной длины — ставит блок данных переменной длины в очередь к указанной в директиве задаче. Максимальный размер буфера — 256(10) слов. Если размер буфера не указан, принимается значение, равное 13(10) словам.

Если в директиве указан флаг события, то событие объявляется при успешном завершении директивы. Указанный флаг устанавливается для посылающей задачи.

Блоки данных переменной длины передаются через вторичный пул. Вызов на языке Фортран:

CALL VSDA (TSK, BUFADR, [BUFLEN], [EFN] [,IDS])

Макровызов:

VSDA$ TSK, BUFADR, [BUFLEN], [EFN]

3. Система управления данными

3.1. НАЗНАЧЕНИЕ СИСТЕМЫ

Система управления данными (СУД) является расширением функциональных возможностей операционных систем СМ ЭВМ в области управления данными. СУД обеспечивает пользователя возможностями создавать, поддерживать и обновлять файлы данных как в пакетном режиме — когда обращение к средствам СУД реализуется через макрокоманды в программе пользователя, так и в интерактивном — когда функции СУД запрашиваются с терминала с помощью обслуживающих программ СУД.

СУД-файл представляет собой набор связанных между собой логических записей. *Логическая запись* — основная единица информации при обработке СУД-файлов. Каждая запись характеризует информационный объект. В рамках одного файла все записи имеют одинаковые характеристики (длина записей может быть различна). Каждый компонент данных занимает некоторую дискретную часть записи, которую называют *полем данных.*

Основная задача СУД заключается в занесении каждой записи, сформированной программой пользователя (ПП), в файл, а также последующего поиска и получения этой записи из файла и ее передача ПП. Структура или тип организации файла определяет способ занесения или получения записи из файла. Метод, по которому программа пользователя обращается к СУД для занесения или считывания записей, называется *методом доступа.* Метод доступа связан с организацией файла.

3.1.1. ОРГАНИЗАЦИЯ ФАЙЛОВ СУД

СУД поддерживает файлы с тремя типами организации: последовательной, относительной и индексно-последовательной (индексной).

Файлы с последовательной организацией. При такой организации записи расположены в файле в порядке их занесения в файл. Добавлять записи можно только в конце файла. Удаление записи или группы записей возможно лишь посредством усечения файла, начиная с записи и до конца. Запись можно корректировать без изменения ее длины. На рис. 2 представлена схематическая структура файла с последовательной организацией.

Рис. 2. Структура файла с последовательной организацией

Файлы с относительной организацией. Такие файлы состоят из ячеек равной длины, которая соответствует максимальной длине записи. Ячейки последовательно пронумерованы от 1 до N. Каждая ячейка может содержать одну запись или быть пустой. Запись может занимать часть или всю ячейку,

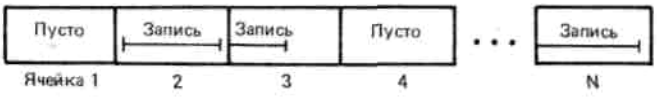

Рис. 3. Структура файла с относительной организацией

но не может быть длиннее ячейки. Запись может быть занесена или извлечена из ячейки с использованием номера ячейки. На рис. 3 представлена структура файла с относительной организацией. Обработка записи такого файла может проводиться независимо от остальных записей.

Файлы с индексной организацией. В записях файлов с индексной организацией выделяются *ключевые поля* (ключи), определяемые пользователем. Позиция и длина ключа должна быть одинаковой для всех записей файла. Значения ключей всех записей файла собраны в специальную таблицу — *индекс.* Для каждой записи в индексе имеются значение ключа и адрес записи, содержащей этот ключ. Значения ключей в индексе упорядочены. Индекс служит для быстрого поиска записей в файле.

Для создания файла с индексной организацией во всех записях должен быть определен, по крайней мере, один ключ. Этот обязательный ключ называется главным, а соответствующий индекс — *главным индексом.* Таким образом, главный индекс содержит ссылки на все записи файла.

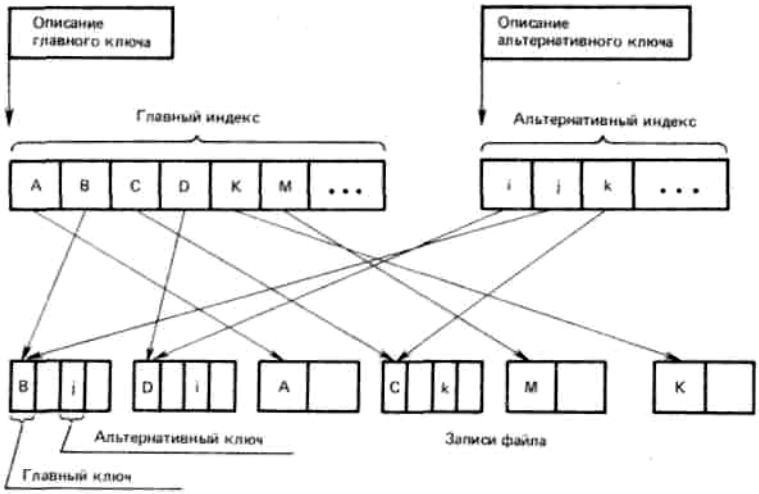

Рис. 4. Структура файла с индексной организацией

Кроме главного ключа в записях может быть определен другой, необязательный ключ, называемый альтернативным. В соответствии с ним существует альтернативный индекс. В альтернативном индексе могут быть ссылки не на все записи файла. Файл с индексной организацией (рис. 4) может содержать один главный и несколько альтернативных индексов. Поиск записей может проводиться по любому индексу.

3.1.2. МЕТОДЫ ДОСТУПА В СУД

Если организация файла выбирается на этапе его создания, то метод доступа к файлу выбирается во время обработки.

СУД предоставляет следующие методы доступа к записи: последовательный, прямой, прямой по адресу записи в файле (АЗФ).

Последовательный доступ может, использоваться для файлов с любой организацией. Тип организации файла определяет порядок, в котором записи заносятся или извлекаются. Для последовательных файлов — это порядок расположения записей в файле, для относительных — порядок расположения ячеек, содержащих записи. Для файлов с индексной организацией порядок обработки записей определяется порядком следования ключей в том индексе, по которому ведется поиск.

Прямой метод доступа позволяет обрабатывать записи в произвольном порядке. Для файлов с последовательной организацией метод допустим, если все записи имеют равную длину и известны номера записей от начала файла. Для относительных файлов это доступ по номеру ячейки, для индексных — по значению ключа.

Метод доступа по физическому адресу записи в файле (АЗФ) допустим для любых файлов на диске. При этом ПП должны указывать для доступа адрес записи в файле, ранее возвращенный в СУД при занесении записи.

Для доступа к виртуальным блокам файла СУД предоставляет последовательный метод и прямой по виртуальному номеру блока.

СУД допускает только те комбинации типов организации файла и методов доступа, которые приведены в табл. 3.1.

Таблица 3.1

Тип организации файла	Метод доступа			
	последова-тельный	прямой		АЗФ
		по номеру	по ключу	
Последовательный	Да	Для файлов с записями фиксированной длины	Нет	Только для файлов на диске
Относительный	Да	Да	Нет	Да
Индексный	Да	Нет	Да	Да

3.1.3. АТРИБУТЫ ФАЙЛОВ И ЗАПИСЕЙ

Каждый СУД-файл имеет логические и физические характеристики, называемые *атрибутами файла.* Часть атрибутов файла хранится в блоке заголовка, а для файлов с относительной и индексной организацией дополнительные атрибуты хранятся в прологе — специально выделяемой части файла.

Создание файлов со структурой СУД с индексными атрибутами можно осуществить либо непосредственно в ПП, либо с помощью обслуживающих программ SUDDES или SUDDEF.

Ниже приводится краткое описание атрибутов файлов и записей.

Тип внешнего устройства — определяет устройство, на котором будет создан файл. Последовательные файлы можно создавать на магнитных лентах и дисках. Последовательные файлы можно выводить на АЦПУ и терминалы. Относительные и индексные файлы могут быть созданы только на дисках.

Спецификация файла и код защиты определяются в СУД в соответствии с соглашениями, принятыми в ОСРВМ.

Организация файла может быть последовательной, относительной или индексной.

Распределение — первоначальный размер файла в блоках.

Увеличение — число блоков, добавляемое к файлу каждый раз при увеличении его размера.

Размер бакета является логической единицей памяти и состоит из одного или нескольких виртуальных блоков, которые СУД воспринимает как единое целое. Записи могут пересекать границы блоков, но не границы бакетов. Размер бакета от 1 до 32 блоков является атрибутом файла и определяется при его создании.

Непрерывность определяет, должны ли файлу выделяться смежные блоки.

Максимальное число записей — для относительных файлов максимальное число записей в файле.

Формат записей — СУД поддерживает следующие форматы записей:

•записи фиксированной длины;

•записи переменной длины;

•записи переменной длины с постоянной управляющей областью (VFC-формат);

• записи поточного формата (только в последовательных файлах), отделяемые друг от друга символами FF, VT, LF, CR/LF.

Формат записи связан с типом устройства и организацией файла в соответствии с табл. 3.2.

Таблица 3.2

Устройство	Организация файла	Метод доступа к записи	Формат записи
Магнитная лента	Последовательная	Последовательный	Фиксированный переменный
Диск	Последовательная	Последовательный АЗФ	Фиксированный переменный VFC-формат поточный
		Прямой по номеру	Фиксированный
	Относительная	Последовательный прямой по номеру АЗФ	Фиксированный переменный VFC-формат
	Индексная	Последовательный прямой по ключу АЗФ	Фиксированный переменный

Размер записи — размер записи в байтах. Для записей переменной длины и поточного формата — это максимальная длина. Для записей фиксированной длины — точная длина записи. Для VFC-записей — максимально допустимая длина переменной части записи.

Размер управляющего поля — для VFC-записей это размер в байтах фиксированной части записи.

Пересечение границ блоков — для последовательных файлов это означает возможность для записей переходить из одного блока в другой.

Номер ключа — последовательный номер, полученный ключом при создании файла: 0 — номер главного ключа; 1 —первый альтернативный ключ и т. д.

Положение ключа — положение ключа в записи.

Размер ключа — длина ключа в байтах.

Тип данных ключа — строка символов, целое число со знаком или без знака, упакованное десятичное число.

Имя ключа — связанная с данным ключом 32-битная последовательность символов КОИ-8.

Повторение значений ключа — возможность дублирования записей с теми же значениями главного или альтернативного ключа.

Изменяемые значения ключей — возможность изменять значения альтернативных ключей.

Нулевые значения ключей — возможность иметь пустые значения ключей в записях.

Сегментированные ключи — возможность использования отдельных полей записи в качестве одного ключа.

Размер заполнения бакета — уровень (в процентах или байтах), до которого должны заполняться записями бакеты.

Область в индексном файле — область представляет собой часть файла, которую СУД трактует как единое целое по отношению к таким характеристикам:

• первоначальное размещение файла;

• расширение файла;

• размер бакета;

• управление размещением на физическом томе.

В различных областях можно размещать разные части различных индексов. Такое разделение приводит к сокращению времени ввода-вывода.

3.1.4. ОБРАБОТКА ФАЙЛОВ, СОЗДАННЫХ СУД

После создания файла ПП могут осуществить доступ к нему для обработки. Операции СУД могут быть разделены на группы:

•операции над каталогами. Эти операции влияют только на записи о файлах в каталогах и не влияют на содержимое файлов;

•операции над файлами. Эти операции обеспечивают доступ к файлам как единому целому, т. е. они не обеспечивают доступа к записям внутри файла;

•операции над потоками. Эти операции являются интерфейсом между программой и записями, которые требуются в пользовательской программе;

•операции над записями. Эти операции обрабатывают записи внутри файла;

•операции над блоками. Эти операции рассматривают файл как последовательность блоков, не анализируя их содержимое.

3.2. ПРОГРАММНЫЙ ИНТЕРФЕЙС

Символы и макрокоманды СУД определяют интерфейс между ПП на языке макроассемблера и подпрограммами СУД. Определения символов и макрокоманд находятся в макробиблиотеке SUDMAC. MLB.

Каждая макрокоманда соответствует конкретной операции СУД и связана с соответствующей подпрограммой. Операция СУД возвращает ПП значение, называемое кодом завершения. Символы завершения СУД присваивают имена кодам завершения. ПП, использующая операции СУД, может обращаться к подпрограммам завершения.

ПП и подпрограммы СУД обмениваются служебной информацией через пользовательские управляющие блоки (ПУБ). Каждый ПУБ имеет имя и состоит из полей, которые также имеют трехбуквенное имя. Ниже приведен список ПУБ.

Имя блока	**Содержание блока**
FAB	Общая информация о файле и о доступе **к** нему
RAB	Информация о потоке, записи или блоке, доступе к записи
ALL	Информация об области размещения файла
DAT	Информация о датах создания и ревизий файла
KEY	Информация о ключе и индексе
NAM	Информация о спецификации файла
PRO	Информация о владельце и защите файла
SUM	Число областей и индексов файла

Для того чтобы употреблять операции СУД в программах на макроассемблере, пользователь должен:

•объявить макрокоманды и символы СУД;

•объявить возможности СУД, используемые ПП;

•объявить и использовать буферный пул, состоящий из пяти буферных пулов конкретного назначения;

•объявить ПУБ и установить в них необходимые значения полей;

•использовать операции СУД, вызывая соответствующие макрокоманды;

•использовать подпрограммы завершения. Применение этих подпрограмм необязательно;

•использовать собственные подпрограммы получения памяти или предоставить СУД возможность применять СУД-подпрограммы получения памяти;

•оттранслировать программу;

•построить задачу. При этом пользователь должен выполнить одно из следующих действий:

использовать резидентную библиотеку СУД;

определить оверлейную структуру СУД для своей задачи;

включить подпрограммы СУД в свою задачу.

3.2.1. ОБЪЯВЛЕНИЕ МАКРОКОМАНД И СИМВОЛОВ СУД

Для извлечения определения макрокоманда СУД ПП может использовать директиву .MCALL в формате

```
.MCALL   список
```

где список — список используемых макрокоманд СУД.

Программа пользователя может употреблять специальные макрокоманды СУД, которые служат для определения символов и макрокоманд.

3.2.2. ОБЪЯВЛЕНИЕ ВОЗМОЖНОСТЕЙ СУД

Возможности СУД, используемые ПП, определяются макрокомандой ORG$. Ф о р м а т

```
MCALL ORG$
ORG$   FILEORG [, <OP1, OP2, ...>]
```

где FILEORG — параметр, определяющий тип организации файла.

Параметр принимает одно из следующих значений: SEQ — при последовательной организации файлов; REL — при относительной организации файлов; IDX — при индексной организации файлов.

При обработке в ПП файлов нескольких (двух или трех) типов организации указывается несколько (две или три) макрокоманд ORG$;

OP1, OP2, ... — список операций, которые будет использовать ПП. Операции задаются списком из указанного ниже множества:

CRE — операция создания файлов;
DEL — операция удаления файлов;
FIN — операция поиска;
GET — операция чтения записей;
PUT — операция занесения записей;
UPD — операция замены записей.

Перечисленные выше операции ПП должна объявлять явно в макрокоманде ORG$. Поддержка других операций обеспечивается автоматически.

3.2.3. ОПИСАНИЕ И ИСПОЛЬЗОВАНИЕ БУФЕРНОГО ПРОСТРАНСТВА

СУД динамически получает и освобождает буферное пространство, которое разделено на пять буферных пулов:

•буферный пул внутренних блоков FAB и блоков описания индекса (IFAB/IRAB);

•буферный пул внутренних блоков RAB (IRAB);

•буферный пул ключей;

•буферный пул ввода-вывода;

•буферный пул блоков описания буферов (BDB).

Пользователь должен описать буферное пространство, используя макрокоманды описания буферного пула. Эти макрокоманды используются в ПП в виде следующего набора:

```
POOL$B
PS FAB FABCOUNT
```

P$IDX INDEXCOUNT
P$RAB RABCOUNT
P$RABX RABXCOUNT, KEYSIZE, KEYCHANGE
P$BUF BUFCOUNT
P$BDB BDBCOUNT
POOL$E

Параметры в этих макрокомандах вычисляются следующим образом:

FABCOUNT = максимальному числу одновременно открытых файлов в ПП;

INDEXCOUNT = максимальному количеству индексов в индексных файлах, обрабатываемых в ПП;

RABCOUNT = максимальному числу потоков доступов (или числу блоков RAB), которые будут одновременно активны в ПП для файлов с последовательной и относительной организацией;

RABXCOUNT = максимальному числу потоков доступов, которые будут одновременно активны для всех индексных файлов в ПП;

KEYSIZE = длине наибольшего ключа в байтах в индексных файлах, обрабатываемых в ПП;

KEYCHANGE = максимальному числу альтернативных ключей, которые могут изменять свои значения во всех обрабатываемых файлах;

BUFCOUNT использует либо частные буфера ввода-вывода для файлов, либо централизованный пул. Параметр BUFCOUNT вычисляется только для файлов, обрабатываемых через централизованный пул. BUFCOUNT = P1 + P2 + ... + PI + ... + PN

Здесь PI — размер в байтах буферов для каждого из потоков доступов:

PI = B∗512∗K — для дисковых файлов, где B — размер бакета файла в блоках;

K — количество буферов для указанного файла;

PI = BL∗K — для файлов на магнитной ленте;

где BL — размер физического блока в байтах;

PI = BS — для файлов других устройств последовательного доступа;

BDBCOUNT = Ml + M2 + (2∗M3)

где Ml — максимальное число буферов, которые будут использовали ПП для всех одновременно открытых файлов;

M2 — максимальное число одновременно активных потоков доступов для относительных файлов;

M3 — максимальное число одновременно активных потоков доступов для индексных файлов.

3.2.4. ОБЪЯВЛЕНИЕ ПУБ И ИНИЦИАЛИЗАЦИИ ПОЛЕЙ ПУБ

Объявление ПУБ и установка их полей выполняются с помощью наборов макрокоманд для каждого типа ПУБ.

Блок FAB создается пользователем для каждого открытого файла:

```
        .EVEN
метка   FAB$B
        .
        .   набор макрокоманд установки полей FAB
        .
        FAB$E
```

Блок RAB создается пользователем для каждого активного потока доступов с параметром SYN — для синхронных операций, с параметром ASYN — для асинхронных операций:

```
        .EVEN
метка:  RAB$B /SYN   \
              \ASYN/
        .
        .   набор макрокоманд установки полей RAB
        .
        RAB$E
```

Блок NAM:

```
        .EVEN
метка:  NAM$B
        .
        .   набор макрокоманд установки полей NAM
        .
        NAM$E
```

Блоки XAB, когда они необходимы, используют наборы макрокоманд с соответствующим параметром в первой макрокоманде:

```
        XB$ALL   — для блока ALL
        XB$DAT   —для блока DAT
        XB$KEY   —для блока KEY
        XB$PRQ   —для блока PRQ
        XB$SUM   —для блока SUM
                . EVEN
                                                  ⎧ XB$ALL ⎫
                                                  ⎪ XB$DAT ⎪
метка:                              XAB$B         ⎨ XB$KEY ⎬
                                                  ⎪ XB$PRQ ⎪
                                                  ⎩ XB$SUM ⎭
                .
                .   } набор макрокоманд установки полей
                .
                XAB$E
```

3.2.5. ОПИСАНИЕ И УСТАНОВКА ПОЛЕЙ ПУБ

В данном подразделе описана структура ПУБ. Для каждого блока приведен список полей, перечисленных в алфавитном порядке имен полей. В описании полей используются следующие общие правила:

1) для каждого поля указывается его трехбуквенное имя:

2) символьное смещение поля образуется в виде O$NNN, если поле состоит из одного байта или слова, и O$NNN0, O$NNN1 и т. д., если поле состоит из нескольких байтов или слов (NNN —имя поля):

3) имена макрокоманд для установки полей ПУБ образуются с помощью имени поля:

F$NNN — макрокоманды установки полей блока FAB;

X$NNN — макрокоманды установки полей блоков ALL, DAT, KEY, PRO, SUM;

R$NNN — макрокоманды установки полей блока RAB;

N$NNN — макрокоманды установки полей блока NAM.

В качестве параметров макрокоманд могут быть числовые значения, адреса и символьные значения;

4) большая часть полей ПУБ, описанных ниже, устанавливается ПП при создании файлов. При обработке существующих файлов некоторые из этих полей устанавливаются СУД. Кроме этих полей, в ПУБ присутствуют поля, значения которых устанавливаются только СУД.

Таблица 3.3

Имя поля	Размер поля	Описание поля и значения параметров в макрокомандах установки
ADD	1 байт	Номер области размещения, описанной блоком ALL. Параметр — число
ALN	1 байт	Указывает выравнивание для области, описанной блоком ALL. Параметры: XB $ LBN — выравнивает на логический блок; XB $ VBN — выравнивает на виртуальный блок
ALQ	2 слова	Содержит размер размещения области, описанной блоком ALL. Параметр — размер области (в блоках)
AOP	1 байт	Указывает непрерывность для области, описанной блоком ALL. Параметры: XB$ HRD — жесткое размещение области; XB$ CTG — непрерывность области
BKZ	1 байт	Содержит размер бакета для данной области. Параметр — размер бакета (в блоках)
DEQ	1 слово	Содержит размера расширения по умолчанию для данной области. Параметр — размер расширения файла (в блоках)
LOC	2 слова	Содержит описание размещения данной области. Значение зависит от параметра ALN: логический номер блока (если XB$ LBN) или виртуальный номер блока (если XB$ VBN). Параметр — номер блока
NXT	1 слово	Содержит адрес следующего блока ALL, DAT, KEY, PRO или SUM при связывании блоков в цепочку. Параметр — символический адрес следующего блока в цепочке

В табл. 3.3 приведено описание блока ALL. Блок содержит информацию об области файла.

В табл. 3.4 приведено описание блока DAT. Блок содержит даты, связанные с файлом, и число ревизий файла.

Таблица 3.4

Имя поля	Размер поля	Описание поля и значения параметров в макрокомандах установки
CDT	4 слова	Содержит двоичную дату создания файла. Дата заносится СУД и представляет собой двоичное число, равное количеству сотен наносекунд, прошедших от даты 00 часов 00 минут
EDT	4 слова	Содержит дату сохранения файла в той же форме, что и поле CDT
NXT	1 слово	Аналогично полю NXT блока ALL
RDT	4 слова	Содержит дату последней ревизии файла в той же форме, что и поле CDT
RVN	1слово	Содержит номер последней ревизии. Заполняется СУД

В табл. 3.5 приведено описание блока FAB. Блок содержит общую информацию о файле и о доступе к нему.

Таблица 3.5

Имя поля	Размер поля	Описание поля и значения параметров в макрокомандах установки
ALQ	2 слова	Содержит размер файла. Параметр — размер файла при его создании
BKS	1 байт	Содержит размер бакета файла. Параметр — размер бакета файла при его создании
BLN	1 байт	Содержит длину» блока FAB в байтах. Устанавливается СУД
BLS	1 слово	Содержит размер блока для последовательного файла на магнитной ленте. Параметр — размер блока в байтах при его создании
BPA	1 слово	Содержит адрес частного буферного пула, если он используется для данного файла, или 0, если используется централизованный пул. Параметр — адрес частного буферного пула
BPS	1 слово	Содержит размер частного буферного пула.
CTX	1 слово	Параметр — размер частного буферного пула в байтах Содержит любую информацию, заносимую в поле ПП. СУД не анализирует содержимое поля
DEQ	1 слово	Содержит размер расширения файла по умолчанию. Параметр — размер расширения в блоках
DEV	1 байт	Указывает характеристики устройства, связанного с файлом. Параметр — символьное значение: FB$CCL — устройство с управляемой характеристикой; FB$ MDI — устройство с многими каталогами; FB$REC — устройство, ориентированное на записи; FB$SDI — устройство с одним каталогом; FB $ SQD — устройство последовательного доступа; FB$TRM — терминал
DNA	1 слово	Содержит адрес строки с именем файла по умолчанию. Параметр — адрес строки
DNS	1 байт	Содержит длину строки с именем файла по умолчанию. Параметр — размер строки в байтах
FAC	1 байт	Указывает запрашиваемый доступ к файлу. Параметр — одно или несколько (связанных знаком конкатенации) символьных значений: FB$ DEL — запрос доступа для операций FIND/GET/DELETE; FB$ GET — запрос доступа для операций FIND/GET; FB$ PUT — запрос доступа для операций PUT; FB $ REA — запрос доступа для операцийREAD; FB$ TRN — запрос доступа для операций FIND/GET/TRUNCATE; FB$UPD — запрос доступа для операций FIND/GET/UPDATE; FB$WRT — запрос доступа для операций ввода-вывода блоков READ/WRITE
FNA	1 слово	Содержит адрес строки с именем файла. Параметр — адрес строки с именем файла.
FNS	1 байт	Содержит размер строки с именем файла в байтах
FOP	1 слово	Указывает признаки вариантов обработки файла; Параметр — одно или несколько символьных значений:

Продолжение

Имя поля	Размер поля	Описание поля и значения параметров в макрокомандах установки
		FB$ RWO — перемотка магнитной ленты перед открытием файла; FB$ RWC — перемотка магнитной ленты после закрытия файла; FB$ POS — позиционирование ленты после последнего закрытого файла; FB$ DLK — запрет блокирования неверно закрытого файла; FB$ CTG — требование непрерывности создаваемого файла; FB$ SUP — перекрытие существующего файла при совпадении спецификаций; FB$ NEF — отказ от позиционирования магнитной ленты на конец файла; FB$ TMP — открываемый файл — временный; FB$ MKD — отметка файла для удаления; FB$ TMD — отметка для удаления временного файла; FB$ FID — использование для открытия файла информации из блока NAM
FSZ	1 байт	Содержит размер постоянной управляющей области для записей VFC-формата. Параметр — размер постоянной управляющей области в байтах, задается при создании файла
LCH	1 байт	Содержит логический номер канала для обработки файла. Параметр — число
LRL	1 слово	Содержит длину максимальной записи в последовательном файле. Значение устанавливается СУД
MRN	2 слова	Содержит максимальное количество записей, допустимое в относительном файле. Параметр — максимальное число записей
MRS	1 слово	Содержит размер записи для записей фиксированной длины или максимальный размер записи для записей других форматов. Параметр — размер записи в байтах
NAM	1 слово	Содержит адрес блока NAM для данного файла. Параметр — символический адрес блока NAM
ORG	1 байт	Содержит тип организации файла. Параметр — одно из символических значений: FB$ IDX — файл с индексной организацией; FB$ REL — файл с относительной организацией; FB$ SEQ — файл с последовательной организацией
RAT	1 байт	Указывает атрибуты записей файла. Параметр — одно или несколько значений: FB$ CR — добавленные LF/CR к выводимой записи; FB$ FTN — запись содержит символ управления кареткой для Фортрана; FB$ PRN — обработка записей типа LF-запись- CR; FB$ BLK — записи не пересекают границы блоков
RFM	1 байт	Содержит код формата записи. Параметр — одно из символьных значений: FB$ FIX — записи фиксированной длины; FB$ STM — записи поточного формата; FB$ UDF — записи неопределенного формата; FB$ VAR — записи переменной длины; FB$ VFC — записи VFC-формата
RTV	1 байт	Содержит размер блока указателя извлечения для файла. Параметр — число указателей извлечения
SHR	1 байт	Указывает совместный разрешенный доступ к файлу. В качестве параметра указывается одно из символьных значений: FB$ GET — совместный доступ для операций FIND/GET; FB$ NIL — запрет совместного доступа; FB$ UPI — совместный доступ любого типа; FB$ WRI — совместный доступ для операций FIND/GET/PUT/UPDATE/DELETE
STS	1 слово	Содержит код состояния завершения операции. Устанавливается СУД
STV	1 слово	Содержит информацию о завершении операции дополнительно к STS. Устанавливается СУД
XAB	1 слово	Содержит адрес первого из блоков, образующих цепочку блоков ALL, DAT, KEY, PRO, SUM. Параметр — адрес первого блока в цепочке

В табл. 3.6 приведено описание блока KEY. Блок содержит информацию о ключе файла и о соответствующем индексе.

Таблица 3.6

Имя поля	Размер поля	Описание поля и значения параметров в макрокомандах установки
DAN	1 байт	Содержит номер области данных индекса, т. е. уровня 0. Параметр — число, номер области
DBS	1 байт	Содержит размер бакета области данных индекса, описанного блоком KEY. Параметр — размер бакета в блоках
DFL	1 слово	Содержит количество байтов для заполнения в каждом бакете области данных индекса, описанного блоком KEY. Параметр — число, количество байтов
DTP	1 байт	Содержит код типа данных ключа, описанного блоком KEY. Параметр — одно из следующих символических значений: XB$ BN2 — 16-битное беззнаковое число; XB$ BN4 — 32-битное беззнаковое число; XB$ 2 — 15-битное целое число со знаком; XB$ IN4 — 31-битное целое число со знаком; XB$ PAC — упакованное десятичное число; XB$ STG — строка
DVB	2 слова	Содержит виртуальный номер блока первого бакета области данных индекса, описанного блоком KEY. Значение устанавливается СУД
FLG	1 байт	Содержит маску характеристик ключа. Параметр — одно или несколько значений: XB$ DUP — разрешено дублирование ключа; XB$ CHG — разрешено изменение ключей при обновлении записей; XB$ NUL — ключи с нулевым значением не включаются в индекс
IAN	1 байт	Содержит номер области, в которой размещены высшие уровни индекса (все, кроме низшего уровня). Параметр — номер области
IBS	1 байт	Содержит размер бакета области индекса, описанного блоком KEY. Значение указывается СУД
IFL	1 слово	Содержит количество байтов, используемых в каждом бакете области индекса, описанного блоком KEY. Параметр — число байтов
KNM	1 слово	Содержит адрес 32-байтового поля, в котором содержится информация, связанная с именем ключа. Параметр — адрес буфера
LAN	1 байт	Содержит номер области, в которой размещен низший уровень индекса, описанного блоком KEY. Параметр — номер области
LVL	1 байт	Содержит количество уровней (не включая уровень данных) индекса, описанного блоком KEY. Значение указывается СУД
MRL	1 слово	Содержит длину максимального размера записи, которую она должна иметь, чтобы полностью поместить ключ, описанный блоком KEY, Значение указывается СУД
NSG	1 байт	Содержит количество сегментов ключа, описанного блоком KEY. Значение указывается СУД
NUL	1 байт	Содержит символ нулевого значения ключа. Параметр — символ или соответствующее ему двоичное число
NXT	1 слово	Содержимое аналогично полю NXT блока ALL
POS	8 слов	Содержит начальные позиции всех сегментов ключа (первой позицией является позиция 0). Параметр — список чисел, разделенных запятыми; каждое число — начальная позиция сегмента ключа
REF	1 байт	Содержит порядковый номер ключа, описанного блоком KEY. Параметр — номер ключа
RVB	2 слова	Содержит виртуальный номер блока, в котором находится корень индекса. Значение устанавливается СУД
SIZ	8 байтов	Содержит размеры всех сегментов ключа. Параметр — список чисел, разделенных запятыми: каждое число — длина соответствующего сегмента ключа
TKS	1 байт	Содержит общий размер ключа (сумма размеров всех сегментов). Значение указывается СУД

В табл. 3.7 приведено описание блока NAM, который содержит специальную информацию об устройстве, каталоге и спецификации файла, а также о символах групповой операции.

Таблица 3.7

Имя поля	Размер поля	Описание поля и значения параметров в макрокомандах установки
DID	3 слова	Содержит идентификатор каталога для заданного файла. Значение устанавливается СУД
DVI	2 слова	Содержит идентификатор устройства для заданного файла. Значение устанавливается СУД
ESA	1 слово	Содержит адрес буфера расширенной строки. Параметр — адрес буфера
ESL	1 байт	Содержит длину расширенной строки. Значение устанавливается СУД
ESS	1 байт	Содержит длину буфера расширенной строки. Параметр — длина буфера в байтах
FID	3 слова	Содержит идентификатор файла. Значение устанавливается СУД
FNB	1 слово	Указывает, какие части полной спецификации (файла были взяты из строки файла и какие — из строки по умолчанию. Значение устанавливается СУД
RSA	1 слово	Содержит адрес буфера результирующей строки. Параметр — адрес буфера
RSL	1 байт	Содержит длину результирующей строки. Значение устанавливается СУД
RSS	1 байт	Содержит размер буфера результирующей строки. Параметр — размер буфера в байтах
WCC	1 слово	Содержит контекст групповой операции. Значение устанавливается СУД
WDI	1 слово	Содержит информацию контекста каталога, заданного символом групповой операции. Значение устанавливается СУД

В табл. 3.8 приведено описание блока PRO, который содержит информацию о владельце файла и защите файла.

Таблица 3.8

Имя поля	Размер поля	Описание поля и значения параметров в макрокомандах установки
NXT	1слово	Содержимое соответствует полю NXT блока ALL
PRG	1слово	Содержит номер члена группы в UIC владельца файла. Параметр — число
PRJ	1слово	Содержит номер группы в UIC владельца файла. Параметр — число
PRO	1слово	Содержит код защиты файла в двоичном виде. Параметр — двоичное число

Таблица 3.9

Имя поля	Размер поля	Описание поля и значения параметров в макрокомандах установки
BKT	2 слова	Содержит виртуальный номер блока для операций ввода-вывода блоков или относительный номер записи для операций последовательного доступа к относительному файлу. Параметр — номер блока или записи
CTX	1 слово	Содержит информацию, аналогичную полю CTX блока FAB
FAB	1 слово	Содержит адрес блока FAB, который необходимо, связать с блоком RAB для создания потока доступов к записям. Параметр — адрес блока FAB
ISI	1 слово	Содержит внутренний идентификатор потока для установления связи блока RAB с соответствующей внутренней управляющей структурой СУД. Значение устанавливается СУД
KBF	1 слово	Содержит адрес буфера ключа для поиска записи во время операции прямого доступа. Параметр — адрес буфера
KRF	1 байт	Указывает на индекс, который используется при выполнении операции. Параметр — порядковый номер ключа или индекса
KSZ	1 байт	Содержит размер ключа записи для операции. Параметр — длина ключа в байтах
MBC	1 байт	Содержит мультиблочный коэффициент (количество блоков последовательного файла на диске для одной операции ввода-вывода). Параметр — количество блоков

Продолжение

Имя поля	Размер поля	Описание поля и значения параметров в макрокомандах установки
MBF	1 байт	Содержит мультибуферный коэффициент (количество буферов для потока доступов операции ввода-вывода). Параметр — количество буферов
RAC	1 байт	Содержит код метода доступа для операций обработки записей. Параметр — одно из символьных значений: RB$ KEY — доступ по ключу; RB$ RFA — доступ по адресу записи; RB$ SEQ — последовательный доступ
RBF	1 слово	Содержит адрес буфера записи для операций над записями. Параметр — адрес буфера
RFA	3 слова	Содержит адрес записи в файле для операций с прямым доступом. Значение устанавливается СУД
RHB	1 слово	Содержит адрес буфера управляющей области для записей VFC-формата. Параметр — адрес буфера
ROP	1 слово	Содержит маску, определяющую вариант обработки записей. Параметр — одно или несколько символьных значений: RB$ ASY —запрашивается асинхронная операция; RB$ EOF — устанавливается позиционирование на конец файла; RB$ FDL — устанавливается процедура быстрого удаления записей; RB$ KGE — указывает на использование в операциях поиска или на получение критерия выбора по ключу «больше или равно»; RB$ KGT — указывает на использование в операциях поиска или на получение критерия выбора по ключу «больше»; RB$ LOA — указывает на соблюдение степени заполнения бакета; RB$ LOC — указывает на использование режима указания места при выполнении операций; RB$ MAS — указывает на использование массового занесения записей; RB$ UIF — указывает на обновление, если запись уже существует
RSZ	1 слово	Содержит размер записи в байтах. Параметр — размер записи
STS	1 слово	Содержит код состояния завершения для операций над записями. Значение устанавливается СУД
STV	1 слово	Содержит информацию, дополняющую поле STS. Значение устанавливается СУД
UBF	1 слово	Содержит адрес пользовательского буфера записей. Параметр — адрес буфера
USZ	1 слово	Содержит размер пользовательского буфера записей. Параметр — размер буфера в байтах

В табл. 3.9 представлено описание блока RAB, который содержит общую информацию о потоке доступов к записям.

В табл. 3.10 описан блок SUM, который содержит количество областей и индексов в данном файле, а также номер версии, который СУД берет из пролога файла.

Таблица 3.10

Имя поля	Размер поля	Описание поля и значения параметров в макрокомандах установки
NOA	1 байт	Содержит количество областей файла. Значение устанавливается СУД
NOK	1 байт	Содержит количество ключей, определенных для индексного файла. Значение устанавливается СУД
NXT	1 слово	Содержимое соответствует полю NXT блока ALL
PVN	1 слово	Содержит номер версии пролога для файла. Значение устанавливается СУД

3.2.6. ИСПОЛЬЗОВАНИЕ ОПЕРАЦИЙ СУД

Макрокоманды установки полей ПУБ заносят информацию на этапе трансляции. Для обработки полей на этапе выполнения ПП предусмотрен специальный набор макрокоманд.

Макрокоманда $STORE предназначена для записи значения в поле ПУБ:

$STORE зн, имп, рег

где зн — адрес в памяти значения, которое должно быть записано в ПУБ;
имп — трехбуквенное имя поля в ПУБ;
рег —регистр (Rl — R5), содержащий адрес ПУБ, в поле которого размещена информация.

Макрокоманда $SET предназначена для установки в единицу битов в однобайтовом или однословном поле ПУБ:

$SET маска, имп, рег

где маска — символьное значение, содержащее биты, которые необходимо установить;
имп — трехбуквенное имя поля ПУБ;
рег —регистр (Rl — R5), содержащий адрес соответствующего ПУБ.

Макрокоманда $OFF предназначена для очистки битов в однобайтовом или однословном поле ПУБ:

$OFF маска, имп, рег

где маска — символьное значение, содержащее очищаемые биты;
имп — трехбуквенное имя поля;
рег — регистр (Rl — R5), содержащий адрес соответствующего ПУБ.

Макрокоманда $ FETCH предназначена для копирования значения из поля ПУБ в буфер пользователя:

$ FETCH — буфер, имп, рег

где буфер — адрес буфера пользователя, куда помещается содержимое поля ПУБ;
имп — имя поля ПУБ;
рег —регистр (Rl —R5), содержащий адрес ПУБ.

Макрокоманда $COMPARE предназначена для сравнения содержимого однобайтового или однословного поля ПУБ с указанным значением:

$COMPARE зн, имп, рег

где зн — адрес в памяти значения, сравниваемого с содержимым поля ПУБ:
имп — имя поля ПУБ;
рег — регистр (Rl—R5), содержащий адрес ПУБ.

Макрокоманда $TESTBITS предназначена для тестирования значений битов в однобайтовом или однобитовом поле ПУБ:

$TESTBITS маска, имп, рег

где маска — символьное значение, содержащее биты, подлежащие тестированию;
имп — имя поля ПУБ;
рег — регистр (Rl — R5), содержащий адрес ПУБ.

Для выполнения системой управления данными требуемых операций ПИБ должны быть связаны между собой:

блок FAB — с блоками RAB путем занесения адреса блока FAB в поле FAB блока RAB;

блок NAM — с блоком FAB путем занесения адреса блока NAM в поле NAM блока FAB;

блок FAB — с цепочкой блоков ALL, DAT, KEY, PRO, SUM путем занесения в поле XAB блока FAB адреса первого блока из цепочки.

Подпрограммы операций СУД вызываются при помощи макрокоманд. Аргументы конкретной подпрограммы операции пользователь может определять двумя способами:

•указать аргументы в макрокоманде:

•разместить аргументы в памяти.

Вызов подпрограмм операций (кроме $RENAME) с использованием макрокоманды с аргументами осуществляется следующим образом:

$MMMMM BLKADR [,[ERRADR] [,SUCADR]]

где $MMMMM — имя макрокоманды (кроме $RENAME);
BLKADR — адрес блока FAB (для операций над файлами или над каталогами) или блока RAB (для операций над потоками доступов, записями или виртуальными блоками);

ERRADR — адрес подпрограммы ошибочного завершения операций;
SUCADR — адрес подпрограммы успешного завершения.

Для вызова подпрограммы операций с размещением аргументов в памяти необходимо:

•создать в памяти блок аргументов;

•занести адрес блока аргументов в регистр R5;

•использовать макрокоманду операции без параметров.

Блок аргументов имеет следующую структуру:

байт 0 — содержит счетчик аргументов;

байт 1 — содержит 0;

байты 2,3 — содержат адрес блока FAB или RAB;

байты 4,5 — содержат адрес подпрограммы ошибочного завершения (необязательный аргумент);

байты 6,7— содержат адрес подпрограммы успешного завершения (необязательный аргумент);

байты 8,9 — содержат адрес нового блока FAB (только для операции RENAME).

Счетчик аргументов может принимать значения 1, 2, 3, 4 в зависимости от числа используемых аргументов. Если аргумент не используется, его значение устанавливается равным —1.

Ниже приводится список макрокоманд СУД:

$CLOSE — для закрытия файла;
$CONNECT — для создания потока доступов к записям открытого файла;
$CREATE — для создания нового файла и открытия его для обработки;
$DELETE — для удаления записи из относительного или индексного файла;
$DISCONNECT — для прекращения потока доступов к записям и отсоединения его от файла;
$DISPLAY — для занесения СУД информации о файле в поля ПУБ. Операция $DISPLAY не меняет файлы;
$ENTER — для включения имени файла в каталог;
$ERASE — для удаления файла и стирания записи о нем из каталога;
$EXTEND — для расширения открытого файла;
$FIND — для поиска записи в файле;
$FLUSH — для записи содержимого буфера ввода-вывода на диск;
$FREE — для освобождения заблокированного файла;
$GET — для передачи записи из файла в буфер ввода-вывода или буфер пользователя;
$NXTVOL — для установки и монтирования следующего тома магнитной ленты;
$OPEN — для открытия файла;
$PARSE — для анализа спецификаций файла;
$PUT — для занесения новой записи в файл;
$READ — для чтения виртуальных блоков;
$REMOVE — для удаления записи о файле из каталога;
$RENAME — для изменения записи о файле в каталоге. Ф о р м а т макрокоманды:

$RENAME OLDFAB, [ERRADR], [SUCADR], NEWFAB

где OLDFAB — адрес блока FAB для операции;
ERRADR — адрес подпрограммы ошибочного завершения; SUCADR — адрес подпрограммы успешного завершения;
NEWFAB — адрес блока FAB, определяющего новую спецификацию файла;

$REWIND — для внутреннего позиционирования на начало файла (для последовательных и относительных файлов) или на начало индекса для индексных файлов;
$SEARCH — для поиска записи о файле в каталоге;
$SPACE — для перемотки магнитной ленты;
$TRUNCATE — для удаления части последовательного файла от текущей записи до конца файла;
$UPDATE — для замены записи в файле на запись, подготовленную ПП;
$WAIT — для задержки (ожидания) выполнения ПП до завершения операции ввода-вывода;
$WRITE — для записи в файл виртуального блока.

3.3. ОБСЛУЖИВАЮЩИЕ ПРОГРАММЫ СУД

Обслуживающие программы СУД позволяют создавать, обрабатывать и поддерживать файлы структуры СУД. Обслуживающие программы являются независимыми задачами, выполняемыми в

интерактивном режиме. Они предназначены для специалистов, которые не пользуются программным интерфейсом СУД. Обслуживающие программы СУД обладают ограниченными возможностями по сравнению с программным интерфейсом.

3.3.1. ПРОГРАММА ПРОЕКТИРОВАНИЯ СУД-ФАЙЛОВ SUDDES

Программа проектирования файла SUDDES дает возможность пользователю проектировать и создавать файлы структуры СУД. Проектирование файла осуществляется путем определения атрибутов файла. Файл создается в соответствии с этими атрибутами.

При проектировании файла пользователь задает команды установки, сброса, очистки или печати значений атрибутов из буфера проектирования. Это называется *сеансом проектирования.* Буфер проектирования — рабочее пространство программы SUDDES; он разделен на пять секций: системы, файла, записей, ключей и областей.

В начале сеанса атрибуты файла устанавливаются в соответствии со значениями, принятыми в СУД по умолчанию. Пользователь может установить атрибуты в интерактивном режиме с терминала командой SET либо из внешнего (существующего файла описания и файла данных) файла — командой GET.

Атрибуты могут быть распечатаны с помощью команды SHOW.

Ниже приводится описание команд программы SUDDES. Команды могут вводиться целиком или последовательно по частям с соответствующими подсказками.

Команда CLEAR в буфере проектирования устанавливает значения атрибутов, принятые в СУД по умолчанию. Ф о р м а т :

CLEAR { ALL | секция ALL | секция атрибут }

где секция — восстанавливает значения по умолчанию всех атрибутов во всех секциях;
секция ALL — восстанавливает значения по умолчанию всех атрибутов указанной секции;
секция атрибут — восстанавливает значения по умолчанию указанного атрибута указанной секции.

Список имен секций и атрибутов приведен ниже.

Команда CREATE создает пустой файл данных, атрибуты которого имеют значения, определенные в буфере проектирования. Ф о р м а т :

CREATE [спецификация файла]

где спецификация файла — спецификация пустого файла данных с атрибутами, определенными в буфере проектирования. Если спецификация не указана в команде, файл данных будет иметь спецификацию, указанную в буфере проектирования. Если и там спецификация не указана, будет создан файл FILE.DAT.

Команда CTRL/Z завершает работу SUDDES, не запоминая описания файла и не создавая пустого файла данных. Ф о р м а т :

<CTRL/Z>

Команда ESC, введенная в ответ на любой запрос-подсказку, печатает на экране терминала подсказку DES:, при этом все значения атрибутов в буфере проектирования сохраняются. Ф о р м а т :

<ESC>

Команда EXIT запоминает атрибуты файла в файле описания, указанном в командной строке. Затем она завершает работу программы SUDDES. Ф о р м а т :

EXIT имя файла [.тип]

где имя файла [.тип] — является именем и типом файла проектирования, в котором запоминается описание файла. Тип файла по умолчанию — DES.

Команда GET устанавливает значения атрибутов в буфере проектирования на основании информации, считанной из внешнего файла. Ф о р м а т :

GET имя файла [. расширение] [тип]

где имя файла [. расширение] — имя и тип внешнего файла. Расширение по умолчанию — DES;
тип — тип создаваемого файла. Если внешний файл является файлом данных, тип по умолчанию совпадает с типом внешнего файла, в противном случае значение по умолчанию — DAT.

Команда HELP выводит на терминал текст, поясняющий работу программы SUDDES. HELP вводится только на подсказку DES. На любой другой запрос-подсказку вводится символ «?». Ф о р м а т :

HELP { COMMANDS | команда | SECTIONS | секция | секция атрибут }

где COMMANDS — перечисляет все команды, для которых имеются поясняющие тексты;
команда — название команды, для которой нужен поясняющий текст;
SECTIONS — перечисляет все секции и дает инструкции по распечатке сообщений о секциях;
секция — название секции, для которой нужен поясняющий текст;
секция атрибут — указывает секцию и атрибут, для которого необходимо распечатать поясняющий текст.

Команда QUIT завершает работу SUDDES, не создавая файла проектирования или пустого файла данных. Ф о р м а т :

QUIT

Команда SAVE запоминает описание проектируемого файла в файле описания, спецификация которого указана в командной строке. Ф о р м а т :

SAVE имя файла [.расширение]

где имя файла [.расширение]— спецификация файла описания, в котором запоминается описание файла. Расширение по умолчанию DES.

Команда SET устанавливает значения атрибутов в буфере проектирования. Пользователь может задать значения для отдельных атрибутов различных секций. Ф о р м а т :

SET { ALL | секция ALL | секция атрибут значение }

где ALL — указывает, что требуется установка значений всех атрибутов во всех секциях. В этом случае SUDDES будет запрашивать о значениях атрибутов в режиме диалога;
секция ALL — указывает, что требуется установка значений всех атрибутов в указанной секции;
секция атрибут значение — задает присвоение данного значения атрибуту, указанному в секции.

Команда SHOW распечатывает значения атрибутов. Пользователь может распечатать значения отдельного атрибута, значения атрибутов одной секции или всех секций. Ф о р м а т :

SHOW { ALL | секция ALL | секция атрибут }

где ALL — распечатывает значения атрибутов во всех секциях;
секция ALL — распечатывает значения всех атрибутов в указанной секции;
секция атрибут — распечатывает значение указанного атрибута в указанной секции.

Команда SHOW ID печатает текущую версию программы SUDDES. Ф о р м а т :

SHOW ID

Буфер проектирования программы SUDDES разделен на пять секций:

секцию системы SYSTEM. В данной секции пользователь определяет целевую операционную систему, в среде которой будет применяться создаваемый файл, а также другие атрибуты. Ниже описываются атрибуты данной секции.

FILE PLACEMENT { YES | NO }

где YES — указывает на то, что атрибут POSITION в секции областей будет определяться;
NO — указывает на то, что атрибут POSITION в секции областей определяться не будет;
TARGET аргумент — определяет рабочую операционную систему, в среде которой файл будет создаваться и

обрабатываться. В качестве аргумента указывается одно из значений:

ORV — для ОСРВМ;

DSKP — для ДОС КП.

По умолчанию устанавливается та система, в которой файл проектируется.

SOURCE — этот атрибут задается программой SUDDES, автоматически устанавливающей ту операционную систему, в которой производится проектирование файла;

секцию файла FILE. Секция имеет следующие атрибуты:

NAME строка — задает имя файла. В качестве строки указывается спецификация файла. По умолчанию FILE.DAT;

ORGANIZATION аргумент — задает тип организации файла. В качестве аргумента следует указать одно из значений:

SEQUENTAL — задает последовательный файл;

RELATIVE — задает относительный файл;

INDEXED — задает индексный файл;

ALLOCATION число — задает начальный размер файла в блоках;

EXTENTION число — указывает количество блоков, которое должно добавляться к файлу при каждом расширении;

BUCKET_SIZE число — указывает количество блоков в бакете для файла;

PROTECTION строка — указывает код защиты файла. Он представляет собой строку символов, которая определяет виды доступа для всех категорий пользователей. Категории и виды доступа указываются в форме

(SYSTEM: RWED, OWNER: RWED, GROUP: RWED, WORLD: RWED)

OWNER строка — указывает UIC владельца файла. По умолчанию берется UIC, под которым выполняется SUDDES;

MAGTAPE_BLOCK_SIZE число — указывает размер физического блока в байтах для файлов на магнитной ленте. По умолчанию — 512 байт;

REWIND_MAGTAPE { YES / NO } — указывает, что магнитная лента должна быть перемотана, и файл будет располагаться в начале ленты.

Здесь YES — указывает на необходимость перемотки;

NO — ленту перематывать не требуется. Это значение по умолчанию;

MAX_RECORD_NUMBER число — указывает максимальное число записей для относительного файла. По умолчанию берется 0, что означает отсутствие контроля на максимальный номер;

CONTIGUOUS { YES / NO } — указывает, должен ли файл быть непрерывным, по умолчанию — NO;

SUPERSEDE { YES / NO } — указывает, должна ли текущая версия заменить более раннюю версию файла с той же спецификацией. Значение по умолчанию — NO, т. е. при совпадении спецификаций будет создан новый файл;

секцию записей RECORD. Эта секция содержит следующие атрибуты:

SIZE число — указывает размер записи в байтах;

FORMAT аргумент — указывает формат записей. В качестве аргумента Должно быть введено:

VARIABLE — для записей переменной длины;
STREAM — для записей поточного формата;
FIXED — для записей фиксированной длины;
VFC — для записей VFC-формата.

Значение по умолчанию — VARIABLE;

CONTROL_FIELD_SIZE число — указывает размер в байтах фиксированной управляющей области для записей VFC-формата:

BLOCK_SPAN { YES / NO } — указывает, могут ли записи пересекать границы блоков;

CARRIAGE_CONTROL аргумент — указывает, как записи выводятся на устройство с управляемой кареткой. В качестве аргумента указывается одно из значений:

CARRIAGE_RETURN — указывает, что каждой записи предшествует символ LF, а за записью размещен CR;

FORTRAN — указывает, что первый байт записи интерпретируется как управляющий символ Фортрана;

NONE — нет управления кареткой;

секцию ключей — KEY N. Секция позволяет определить ключи индексного файла. При проектировании файла пользователю задается вопрос о количестве ключей в файле: «NUMBER OF KEYS?.» При ответе необходимо указать

число ключей файла. Данная секция будет разбита на подсекции в соответствии с числом ключей. Каждая подсекция имеет имя KEY N, где N — порядковый номер ключа: 0 — главный; 1 — первый альтернативный и т. д. Каждая подсекция имеет следующие атрибуты:

NAME строка — указывает имя ключа, состоящее не более чем из 32 символов КОИ-8;
TYPE аргумент — указывает тип данных в поле ключа. Аргумент принимает одно из следующих значений:
STRING — значение ключа. Строка символов;
BIN2 — значение ключа. 2-битовое целое число без знака;
BIN4 — значение ключа. 4-битовое целое число без знака;
INT2 — значение ключа. 2-битовое целое число со знаком;
INT4 — значение ключа. 4-битовое целое число со знаком;
DECIMAL — значение ключа. Упакованное целое десятичное число;

NUL_KEY { YES / NO } — указывает, будет ли файл содержать нулевое значение ключа в некоторых записях;

NULL_VALUE аргумент — указывает байт нулевого значения. В качестве аргумента указывается:
символ КОИ-8;
десятичное число в диапазоне от 0 до 255.
Значение по умолчанию — пробел;

DUPLICATES логическое значение — указывает, может ли дублироваться значение ключа в индексе. Указывается логическое значение:

YES — дублирование допустимо;
NO — дублирование запрещено;

CHANGES логическое значение — указывает, может ли изменяться значение ключа при модификации записи. Указывается логическое значение:

YES — изменение допустимо;
NO — изменение запрещено;

DATA_FILL число — указывает процент заполнения бакетов уровня данных при начальной загрузке файла;
DATA_AREA число — указывает номер области, которая будет содержать записи данных главного ключа или указатели на записи данных для альтернативного ключа;
INDEX_FILL число — указывает процент заполнения бакетов индексного уровня при начальной загрузке файлов;
LEVEL1_INDEX_AREA число — указывает номер области, которая будет содержать низший уровень индекса данного ключа;
INDEX_AREA число — указывает номер области, которая будет содержать верхние уровни индекса для данного ключа.

Строковые ключи могут иметь более одного сегмента. Для строковых ключей задается вопрос о числе сегментов, из которых состоит ключ: «NUMBER OF SEGMENTS?». Для каждого сегмента должны быть определены:
SEGN_POSITION число — указывает местоположение сегмента в записи, начиная с 0-го байта;
SEGN_LENGTH число — указывает длину сегмента в байтах;
SEGO_POSITION число — указывает местоположение нестрокового ключа;

секцию областей —AREAN N. Секция позволяет определить области индексного файла. Все области нумеруются, начиная с 0. Каждая область описывается своей подсекцией AREA N, где N — номер области. Области могут быть определены по умолчанию, для чего пользователю задается вопрос: «AREAS DEFAULTED:». На вопрос следует ответить YES или NO.

При явном определении областей пользователю задается вопрос об их количестве: «NUMBER OF AREAS:». Ответом является число областей.

Для каждой области описываются следующие атрибуты:

ALLOCATION число — указывает начальный размер области в блоках. По умолчанию 0 блоков;
EXTENTION число — указывает количество блоков, которое добавляется к области при ее расширении;
BUCKET_SIZE число — указывает количество блоков в бакете для области;

CONTIGOUS { YES / NO } — указывает, должна ли область состоять из непрерывной части дискового пространства;

POSITION аргумент — указывает тип размещения области. В качестве аргумента должно быть задано:
NONE — область размещения на свободном месте диска;
VIRTUAL число — область должна начинаться с указанного пользователем виртуального блока;
LOGICAL число — область должна начинаться с указанного пользователе логического блока;

EXACT_POSITIONING { YES / NO } — указывает, должна ли область размещаться точно с заданного логического блока или возможно ближе к заданному логическому блоку.

3.3.2. ПРОГРАММА ЗАПОЛНЕНИЯ ПУСТОГО ИНДЕКСНОГО ФАЙЛА SUDFIL

Обслуживающая программа SUDFIL позволяет быстро и эффективно заполнять пустой индексный файл записями. Программа может считывать записи из СУД-файла любого типа и загружать их в созданный индексный файл.

Программу можно также использовать для «сжатия» индексных файлов и для преобразования файлов с поточными записями в индексные файлы. Ф о р м а т :

выводной файл [/ключ ...] = вводной файл '[/ключ ...]

где выводной файл — спецификация существующего выводного файла. Он должен отвечать следующим требованиям:
выводной файл должен существовать и в нем не должно быть записей;
он должен иметь индексную организацию;
размер бакета не должен превышать 5 блоков;
в выводном файле не должно быть определено свыше 20 ключей;
вводной файл — спецификация вводного файла, который является источником записей.

К л ю ч и выводного файла программы SUDFIL:

/ER [:спецификация файла] — программа SUDFIL продолжает обработку вводного файла, даже если в нем встречена исключенная запись.

Если указана спецификация файла, в него будут заноситься исключенные записи;

/LO — программа SUDFIL заполняет бакеты в соответствии с величинами заполнения, установленными при создании файла. По умолчанию бакеты заполняются полностью;

/NOER [:S]—программа печатает сообщение об ошибке и заканчивает обработку, если встречает запись, которая не может быть помещена в выводной файл. Если не задан аргумент S, выводной файл удаляется. Если задан аргумент S, выводной файл сохраняется;

/PD [:[#]X] — программа дополняет записи вводного файла до длины записей выводного, если записи выводного файла фиксированной длины;

PD — записи дополняются нулями;
PD:X — дополняются символами X из КОИ-8;
PD:#X— дополняются восьмеричным числом X;

/TR — программа усекает записи вводного файла, если он больше записей выводного файла фиксированного формата.

К л ю ч и в в о д н о г о ф а й л а :

/DE:DEV1 [: DEV2...: DEV5] — программа переназначает устройства, на которых будут располагаться три рабочих файла сортировки и два временных файла для главного и альтернативного ключей:

/KR:N — программа считывает вводной индексный файл в соответствии с ключом, определенным номером N, где N = 0 для главного ключа, N = 1 для первого альтернативного ключа и т.д.;

/NOSO — вводной файл отсортирован.

3.3.3. ПРОГРАММА ПЕРЕСЫЛКИ ЗАПИСЕЙ МЕЖДУ ФАЙЛАМИ SUDCNV

Программа SUDCNV обеспечивает:

•создание нового файла с последовательной организацией, содержащего записи из существующего файла с последовательной, относительной или индексной организацией;

•замену содержимого существующего файла с последовательной организацией записями другого файла с последовательной, относительной или индексной организацией;

•добавление к записям последовательного файла записей другого файла с последовательной, относительной или индексной организацией;

•занесение в существующий файл с относительной или индексной организацией записей другого файла с последовательной, относительной или индексной организацией.

В операционных системах с сетевой поддержкой программа SUDCNV может работать с удаленными файлами. Ф о р м а т :

выводной файл [/ключ...] = вводной файл [/ключ...]

где выводной файл — спецификация выводного файла. В этом файле помещаются записи вводного файла. Если выводной файл имеет относительную или индексную организацию, они должны быть созданы заранее;

вводной файл — спецификация вводного файла.

Г л о б а л ь н ы е к л ю ч и , поддерживаемые программой SUDCNV:

/ID — позволяет получить номер текущей версии SUDCNV; /SL [: спецификация файла] — SUDCNV выдает протокол работы на терминал или в указанный файл.

Локальные ключи программы SUDCNV:

/AP — программа SUDCNV добавляет записи к существующему последовательному файлу;

/BL [: NNNN]— устанавливается размер физического блока для выводного последовательного файла на магнитной ленте. По умолчанию — 512 байт;

/CA '[: спецификация файла] — SUDCNV создает выводной файл с атрибутами существующего файла и копирует в него записи вводного файла. Если в ключе задана спецификация файла, выводной файл создается с этой спецификацией;

/EO — преобразует символы конца файла (CTRL/Z) в потоковых файлах в нулевой байт и заполняет оставшуюся часть файла этими байтами;

/FO: аргумент — ключ определяет тип организации выводного файла. Аргумент принимает одно из значений:

S — последовательная организация (по умолчанию);

R — относительная организация;

I — индексная организация;

/IM — записи вводного файла записываются в выводной блоками;

/KN: ["]ключ["]— записи читаются из вводного индексного файла в соответствии с ключом, имя которого указано;

/KR:N — записи читаются из вводного индексного файла в соответствии с ключом, номер которого N;

/LO — бакеты выводного файла с индексной организацией заполняются в соответствии с размером заполнения бакета, заданным при создании файла;

/MA — SUDCNV использует метод ввода-вывода большого количества записей;

/ML:N — при использовании блочного доступа этим ключом задается количество блоков N, обрабатываемых одной операцией READ или WRITE. Для обычных режимов доступа ключ задает размер памяти под буфера, где N — число Кслов, отведенных под буфера;

/PD [: [#]X ["]] — задает байт, дополняющий записи фиксированной длины выводного файла. Значение ключа соответствует:

/PD — байт 000;

/PD:#X — байт X является восьмеричным числом;

/PD: "X" — байт X является символом КОИ-8, символы нижнего регистра и пробел обязательно заключены в кавычки;

/SL [:спецификация файла] — SUDCMV создает протокол работы на терминале или в файле, указанном в ключе;

/SU — выводной файл совпадает по спецификации с уже существующим файлом. Заменяет существующий файл;

/TR — записи вводного файла усекаются до длины записей выводного файла;

/FW — ключ управляет преобразованием записей формата VFC.

3.3.4. ПРОГРАММА ВЫВОДА АТРИБУТОВ СУД-ФАЙЛА SUDDSP

Обслуживающая программа SUDDSP предоставляет пользователю возможность получить:

•список атрибутов одного или группы файлов;

•список файлов, содержащихся в контейнерном файле, созданном программой SUDBCK.

Формат:

[выводной файл =] вводной файл [/ключ...][, вводной файл [/ключ...]...] где выводной файл — спецификация файла листинга, созданного SUDDSP. По умолчанию — вывод из терминала;

вводной файл — СУД-файл, атрибуты которого нужно вывести, или контейнерный файл.

Ключи программы SUDDSP:

/ID — выводит информацию о текущей версии программы SUDDSP; /BP — вводит краткую информацию о контейнерных файлах; /FU — выводит подробную информацию о СУД-файлах или контейнерных файлах.

3.3.5. ПРОГРАММА СОЗДАНИЯ ЗАПАСНЫХ КОПИЙ SUDBCK

Обслуживающая программа SUDBCK создает на магнитной ленте или на диске специальные контейнерные файлы, содержащие в себе один или несколько сохраненных СУД-файлов. Сохраняются данные СУД-файлов и все атрибуты, кроме атрибута размещения.

При использовании программы SUDBCK имеют место следующие ограничения:

•в качестве вводных файлов пользователь может указать только те файлы, которые размещены на диске;

•программа не может скопировать системный диск как загружаемый том;

•когда резервным носителем является диск, пользователь не может переименовать файлы, т. е. имя каждого запасного файла будет совпадать с именем исходного файла;

•если резервный носитель — магнитная лента, она должна иметь метки в ЕС-формате;

•СУД-файлы, находящиеся в контейнерных файлах, не могут обрабатываться обычными средствами СУД.

Формат:

выводной файл [/ключ ...]= вводной файл [,ключ ...] [,вводной файл [/ключ ...]...]

где выводной файл — спецификация файла запасной копии (контейнерного файла). Если носителем является диск, то имя и расширение задаются в виде *.*;

вводной файл — спецификация файла, для которого создается запасная копия. В спецификациях могут указываться звездочки.

Глобальные ключи:

/ID — указывает номер текущей версии программы SUDBCK;

/ [NO] QU— разрешает или отменяет режим опроса. Когда режим опроса разрешен, программа дает возможность продолжить или приостановить обработку в ошибочных ситуациях;

/SL[: спецификация файла] — обеспечивает выдачу протокола работы программы на терминал или в указанный файл.

Ключи для выводного файла:

/RA — контрольное чтение файла-копии после копирования;

/RC — контрольное чтение и сравнение каждого записанного блока файла-копии;

/RW — перемотка ленты перед записью;

/SU — замена существующего файла новым с совпадающей спецификацией. Только для дисков.

Ключи для вводного файла:

/CD:дд-ммм-гг [: аргумент]— копируются только те СУД-файлы , дата создания которых связана с указанной датой. Если аргумент отсутствует, выбираются файлы с указанной датой создания. Значение аргумента «А» определяет выбор тех файлов, которые созданы после указанной даты. Значение аргумента «В» определяет для копирования файлы, созданные до указанной даты;

/RD: дд-ммм-гг [: аргумент]— файлы запасных копий создаются на основе даты последнего обновления исходных файлов. Значения аргумента соответствуют значениям аргумента предыдущего ключа.

3.3.6. ПРОГРАММА ВОССТАНОВЛЕНИЯ ФАЙЛОВ SUDRST

Программа SUDRST восстанавливает файлы, которые были скопированы в контейнерные файлы программой SUDBCK. При восстановлении сохраняются данные и все атрибуты СУД-файлов, кроме размещения файла. Формат:

выводной файл [/ключ] = вводной файл [/ключ...] [,вводной файл [/ключ...]. . .]

где выводной файл — спецификация файла (файлов), подлежащих восстановлению. Имя и расширение указываются звёздочками (*, *);

вводной файл — спецификация файла, созданного программой SUDBCK.

Глобальные ключи:

/ID — идентифицирует текущую версию программы SUDRST;

/[NO]CV — задает или отменяет преобразование номера версии файла из десятичного в восьмеричное представление для файлов МОСВП, переносимых в ОСРВМ;

/[NO]QU — разрешает или отменяет режим опроса;

/SL [: спецификация файла]— выводит протокол работы на терминал или в указанный пользователем файл.

Ключи для выводного файла:

/FR — ключ изменения кода защиты восстанавливаемого файла. Код защиты устанавливается таким, чтобы UIC владельца файла совпадал с UIC, под которым выполняется SUDRST;

/RA — ключ контроля по чтению каждого блока файла;

/RC — ключ контроля по сравнению данных;

/SU — ключ замены существующего файла восстанавливаемым файлом при совпадении их спецификаций.

Ключи для вводного файла:

/BD:дд-ммм-гг — восстановление дается на основе даты копирования файлов;

/OA: [гр, чл] — восстановление файлов, взятых программой из каталога (или каталогов) (гр, чл);

/SE:спецификация файла — ключ указывается для восстановления СУД-файлов, выбираемых из контейнерного файла на магнитной ленте.

3.3.7. ИНТЕРАКТИВНАЯ ПРОГРАММА СОЗДАНИЯ СУД-ФАЙЛОВ SUDDEF

Программа SUDDEF предназначена для создания СУД-файлов в процессе диалога с пользователем.

Атрибуты файла задаются путем ответа на запросы SUDDEF.

Программа SUDDEF позволяет также создавать косвенный командный файл, который может быть использован в дальнейшем для создания и пересоздания файла. SUDDEF только создает новый СУД-файл, но не заносит в него данные. Атрибуты файла, определяемые пользователем в процессе диалога, описаны в разд. 3.3.1.

4. Диалоговый командный язык

Диалоговый командный язык (DCL) обеспечивает пользователя ОСРВМ (и ОСРВ версии 3.2) расширенным по сравнению с программой связи с оператором (MCR) набором команд для управления выполнением программ, работы с устройствами и файлами, интерактивной разработки программ. Команды DCL системы можно использовать в полном и сокращенном форматах. При этом достаточно ввести столько первых символов соответствующего английского слова, сколько необходимо для однозначной идентификации команды, ее параметров и ключей.

В операционную систему ОСРВМ может входить несколько интерпретаторов командных строк (CLI), основные из которых DCL и MCR [*]. В ранних версиях ОСРВ был один интерпретатор — MCR. Основное отличие между ними состоит в том, что в программе MCR, как правило, указывается имя задачи, которая должна выполнять необходимые действия, а в команде DCL указывается само действие. При выполнении команды DCL транслируются системой в команды MCR. Одним из важных преимуществ DCL является то, что его функции включают в себя функции большинства обслуживающих программ ОСРВМ.

Пользователь, работающий за терминалом с MCR, может вводить команды DCL и, наоборот, работая с DCL, может вводить команды MCR. Для этого после подсказки DCL вводятся символы ”MCR” и команда или после подсказки команды MCR вводятся символы ”DCL” и команда.

4.1. КОМАНДНАЯ СТРОКА DCL

Команда DCL состоит из имени команды или глагола, описывающего действия, которые требуется выполнить в системе. В большинство команд входят один или несколько параметров и ключей, уточняющих действие команды. Ключам предшествует косая черта (/), а параметрам — пробел (_). Как ключи, так и параметры могут иметь аргументы, которым предшествует двоеточие (:).

При вводе некоторых команд, когда требуется указание параметров или аргументов для обеспечения работы, DCL выводит подсказку, определяющую сущность запрашиваемого элемента. Для получения более подробной информации можно ввести знак вопроса (?). Например, при вводе команды RENAME (переименовать) оператор не указал имени нового файла:

```
> RENAME FILE1
```

DCL выводит подсказку:

```
NEW FILE NAME? (новое имя файла)
```

В ответ на эту подсказку пользователь должен ввести имя нового файла — FILE2.

Ключи модифицируют действие команды. Ключи являются необязательными элементами и подразделяются на ключи команды и ключи параметров (файловые). Ключи команды используются в виде модификатора или уточняющего слова к глаголу, определяющего действие. В большинстве случаев ключи команды могут появляться в любом месте командной строки (быть плавающими), например

```
>TYPE /TODAY∗.HLP >TYPE ∗.HLP/TODAY
```

В данных случаях команда печати TYPE работает одинаково.

Ключи параметров модифицируют или уточняют действие команды относительно того параметра, к которому относится ключ (они, естественно, плавающими не являются).

Многие ключи имеют отрицательную форму, которая указывается приставкой (NO) или знаком минус (—).

Если в команде DCL задан список параметров, то элементы его должны отделяться друг от друга запятыми. Некоторые команды могут содержать список аргументов, относящихся к некоторому ключу или параметру. Этот список заключается в круглые скобки, а элементы его разделяются запятыми. От ключей или параметров аргументы отделяются двоеточием (:). Двоеточие, предшествующее аргументу, может быть заменено знаком равенства.

Некоторые команды DCL используются в качестве параметров строки символов, вводимых с

[*] Команды программы МС из-за ограниченного объема книги не описываются, для справок можно воспользоваться [1].

терминала. Если вводятся символы русского алфавита и нежелательно их преобразовывать в латинские, строка русских символов должна быть заключена в двойные кавычки (").

При работе в режиме DCL в системе, имеющей связи с удаленными узлами посредством программного обеспечения локальных и распределенных сетей (СПО СЕТЬ СММ), к стандартной спецификации файла в команде DCL добавляется имя удаленного узла, отделенное двойным двоеточием (::). С удаленными узлами могут работать команды APPEND, COPY, CREATE, DIRECTORY, TYPE.

Для переноса на другую строку части команды DCL используется дефис. В этом случае в начале следующей строки выводится подсказка продолжения (→). Если в командной строке DCL используется комментарий, то ему должен предшествовать восклицательный знак (!).

Ошибки, обнаруженные при выполнении команды DCL, вызывают сообщения на терминал, с которого была введена команда.

4.2. ГРУППЫ КОМАНД DCL

Полный набор команд DCL может быть разбит на семь групп. Внутри каждой группы команды делятся на привилегированные и непривилегированные.

Первая группа — команды SET и SHOW. Команда SET — привилегированная, позволяет изменить характеристики почти всех компонентов системы. Команда SHOW — непривилегированная, позволяет вывести на терминал значения запрашиваемых характеристик системы.

Вторая группа команд относится к операциям с терминалом. Эти команды позволяют: осуществлять регистрацию в системе (LOGIN), производить выход из системы (LOGOUT), получать информацию о командах DCL (HELP), посылать сообщения на другие терминалы (BROADCAST) и на консоль (REQUEST), устанавливать и получать характеристики терминала (SET TERMINAL, SHOW TERMINAL).

Третья группа — команды управления файлами. Они служат для создания, объединения, корректировки, удаления, копирования, сравнения файлов и каталогов. Это команды CREATE DELETE, DIRECTORY, EDIT, PURGE, RENAME, COPY и др. К третьей группе относятся также команды работы с очередями: PRINT, SHOW, QUEUE, SET QUEUE, HOLD и др.

Четвертая группа — команды управления устройствами и томами. Сюда относятся команды для назначения и освобождения логических имен устройств (ASSIGN, DEASSIGN), резервирования и освобождения устройств (ALLOCATE, DEALLOCATE), монтирования и демонтирования (MOUNT, DISMOUNT), определения качества тома и инициализации его (ANALYZE/MEDIA, INITIALIZE), копирования томов (BACKUP) и некоторые другие, в основном информационные команды.

Пятая группа — команды разработки программ. Это команды работы с компиляторами, библиотекарем и построителем задач.

Шестая группа — команды управления задачами. Сюда относятся команды запуска, завершения, установки (RUN, ABORT, INSTALL), фиксации в памяти (FIX, UNFIX), команды работы' с разделами, приоритетами, задачами и др.

Седьмая группа — команды управления системой и информацией о системе. Отличие этих команд от команд первой группы состоит в том, что директивы SET и SHOW имеют уточняющие параметры. Часть команд DCL выполняется системными обслуживающими программами, а часть требует обращения к MCR.

4.3. КОМАНДЫ DCL

Команда ABORT приводит к принудительному завершению выполняемой команды или задачи. Привилегированные пользователи могут аварийно завершить любую задачу, непривилегированные — только запущенную со своего терминала. Система завершает задачу в четыре этапа:

1) устанавливает приоритет задачи равным 247, так как задача при завершении должна быть резидентной в памяти;
2) завершает операции ввода-вывода;
3) выполняет задачу TKTN, которая выводит сообщение о завершении задачи на TI:;
4) освобождает зарезервированную за задачей память, если задача не была фиксирована.

Ф о р м а т :

ABORT имя объекта

К л ю ч и :

/COMMAND — указывает прекращение действия команды (значение по умолчанию);
/ [NO] POSTMORTEM — указывает, будет ли выводиться дамп задачи перед ее завершением;
/TASK — указывает на прекращение выполнения задачи. Ключ должен следовать непосредственно за командой;
/TERMINAL:TTNN: — указывает, что будет отменена задача, загруженная с заданного терминала. Это привилегированный ключ.

Команда ALLOCATE объявляет указанное устройство личным. Ф о р м а т :

ALLOCATE DD [NN:] [логическое-имя]

где DD [NN:] — спецификация устройства;
логическое-имя — имя, которое назначается данному устройству.

К л ю ч и :

/TERMINAL:TTNN: — резервирует устройство за указанным терминалом. Ключ привилегированный;
/TYPE:тип — используется для присоединения первого доступного устройства указанного типа.

Команда ANALYZE/CRASH_DUMP форматирует и анализирует последовательный файл распечатки системной памяти. Описание ключей команды и ее работы дано в разд. 6.

Команда ANALYZE/MEDIA служит для идентификации номеров дефектных блоков на диске для последующего их использования командами BACKUP и INITIALIZE. Ф о р м а т :

ANALIZE/MEDIA DDNN:

где DDNN: — устройство, на котором установлен проверяемый том.

К л ю ч и :

/ALLOCATE:метка-тома — запрашивает информацию о дефектных блоках, помещает ее в файл описателя дефектных блоков и дополняет файл [0, 0] BADBLK.SYS;
/BADBLOCKS — запрашивает номера дополнительных дефектных блоков, которые добавляются в файл описателя дефектных блоков, в дополнение к обнаруженным в процессе проверки;
/BADBLOCKS/NOEXERCISE— заносит номера указанных дополнительных дефектных блоков в файл описателя дефектных блоков, но не присоединяет эти блоки к файлу [0,0] BADBLK.SYS, благодаря чему измененный описатель можно будет использовать при новой инициализации тома. Для того чтобы использовать данный ключ, том должен быть монтирован с ключом /FOREIGN;
/OVERRIDE — игнорирует информацию на последней дорожке устройства и создает файл описателя дефектных блоков в последнем недефектном блоке перед последней дорожкой;
/RETRY — указывает, что драйвер ввода-вывода будет перепроверять найденный дефектный блок перед тем, как считать его дефектным;
/SHOW — указывает, что команда будет выводить список обнаруженных дефектных блоков, кроме тех, которые были введены с помощью ключа /BADBLOCKS.

Команда APPEND позволяет добавить к концу последовательного файла содержимое указанных последовательных файлов. Ф о р м а т :

APPEND вводной-файл выводной-файл

К л ю ч и :

/DATE: DD-MMM-YY — определяет, что будут присоединены только те файлы, которые были созданы в указанное время;
/EXCLUDE: спецификация-файла — определяет файлы, которые не участвуют в выполнении команды;
/NOWARNINGS — подавляет вывод сообщений об ошибках в процессе выполнения команды;
/REWIND — вызывает перемотку ленты на начало перед выполнением операции;
/SHARED — указывает, что другие пользователи могут иметь доступ к файлам, пока длится операция;
/SINCE: DD-MMM-YY — указывает, что будут присоединены только те файлы, которые были созданы, начиная с указанного времени;
/THROUGH:DD-MMM-YY — указывает, что будут присоединены только те файлы, которые были созданы до указанной даты;
/TODAY — указывает, что будут присоединены файлы с текущей датой создания.

Команда ASSIGN для системы без расширенных логических имен связывает логическое имя с физическим устройством, псевдоустройством или логическим устройством.

Выбор назначения логического имени устройству осуществляется в следующем порядке: задача, локальное имя, назначение при регистрации в системе, групповое имя, глобальное имя. Ф о р м а т :

ASSIGN DDNN: логическое-имя

где DDNN: — спецификация физического устройства, псевдоустройства или логического устройства, которому ставится в соответствие логическое имя того же формата.

К л ю ч и :

/GLOBAL — указывает, что назначение включается в системную таблицу назначений, т. е. распространяется на все задачи в системе. Ключ привилегированный;

/LOCAL — указывает, что назначение включается в пользовательскую таблицу логических имен, т. е. является локальным для данного терминала;

/LOGIN — указывает, что назначение производится при регистрации в системе. Ключ привилегированный;

/SYSTEM — то же, что и /GLOBAL;

/TERMINAL:TTNN: — указывает, что назначение относится к данному терминалу, который должен быть зарегистрирован в системе. Ключ привилегированный.

Команда ASSIGN для системы с расширенными логическими именами связывает логическое имя с физическим устройством, спецификацией файла или ее частью, с другими логическими именами. Ф о р м а т :

ASSIGN эквивалентное-имя логическое-имя

где эквивалентное-имя — спецификация устройства с файловой структурой или спецификация файла, которому ставится в соответствие логическое имя. Эквивалентное имя может быть также другим логическим именем. Если эквивалентное имя заключено в кавычки, то оно сохраняется. Это отличает команду ASSIGN от команды DEFINE;

логическое-имя — имя, которое ставится в соответствие устройству или спецификации файла.

К л ю ч и :

/FINAL — указывает, что трансляция эквивалентного имени заканчивается на этом значении;

/GLOBAL — то же, что и в предыдущей команде;

/GROUP [:N] — указывает, что значение включается в групповую таблицу логических имен, N — номер группы пользователей, которым доступно это имя. Если N не указан по умолчанию используется номер группы пользователя, выдавшего команду;

/LOCAL, /LOGIN, /SYSTEM, /TERMINAL: TTNN: — то же, что и в предыдущей команде;

/TRANLATION_ATTRIBUTES = TERMINAL —то же, что и /FINAL.

Команда ASSIGN/QUEUE устанавливает связь между очередью и процессором обработки в диспетчере очередей. Подробная информация о команде приведена в разд. 6. Ф о р м а т :

ASSIGN/QUEUE имя-очереди имя-процессора

Команда ASSIGN/REDIRECT направляет вывод с одного физического устройства на другое. Можно направлять вывод с физического устройства на псевдоустройство и наоборот. Ф о р м а т :

ASSIGN/REDIRECT старое-DDNN: новое-DDNN:

где старое-DDNN: — спецификация устройства, с которого требуется перенаправить вывод;

новое-DDNN:— спецификация устройства, на которое перенаправляется вывод.

Команда ASSIGN/TASK переназначает номера логических устройств (LUN) для установленной задачи с одного физического устройства на другое. Данное переназначение отменяет постоянное назначение логических номеров устройств в файле образа задачи на диске. Команду нельзя применять к задачам, резидентным в памяти. Ф о р м а т :

ASSIGN/TASK имя DDNN: LUN

где имя — имя установленной задачи, для которой необходимо переназначить LUN;

DDNN: — спецификация устройства, которому назначается указанный LUN, может быть физическое устройство или логическое имя устройства.

Команда BACKUP сохраняет и восстанавливает тома с файловой структурой ОСВРМ, а также копирует файлы с одного тома на другой. Команда BACKUP запускает программу BRU, информация о которой содержится в разд. 7. Ф о р м а т :

BACKUP вводное-устройство: [спецификация-файла] выводное-устройство:

Описание специфики работы команды и ее ключи соответствуют описанию программы BRU.

Команда BROADCAST выводит указанное сообщение на один или несколько терминалов, используя системную задачу BRO. Для вывода символов нижнего регистра их следует заключить в кавычки (”). Сообщение BRO может прервать любой вид операций ввода-вывода на указанном терминале. Ф о р м а т 1 :

BROADCAST сообщение

где сообщение — текст, длина которого не должна превышать длину строки экрана терминала.

Ключи:

/ALL — сообщение посылается на все терминалы, исключая подчиненные;
/LOGGED IN — сообщение посылается на все терминалы, где зарегистрированы пользователи. Оба ключа привилегированные.

Формат 2:
BROADCAST имя-пользователя сообщение
где имя-пользователя —USERNAME, которому посылается сообщение.

Формат 3:

BROADCAST@ спецификация-косвенного-командного-файла

где спецификация-косвенного-командного-файла содержит сообщение в формате:

TTNN: сообщение

Привилегированный пользователь может создавать командный файл в формате:

ALL: сообщение

или

LOGGED_IN: сообщение

Командный файл может содержать несколько записей указанного формата, но не может содержать никаких директив и меток. Строки в командном файле не должны начинаться со знака табуляции или знака пробела.

Команда CANCEL удаляет элемент из очереди запросов запуска задачи по времени. Формат:

CANCEL имя-задачи

Задача должна быть установлена.

Команда CONTINUE возобновляет выполнение ранее приостановленной задачи. Формат:
CONTINUE [имя-задачи]

Если имя приостановленной задачи не указано, по умолчанию указывается имя текущего терминала.

Команда CONVERT копирует записи из одного файла в другой, используя программу SUDCNV, описанную в гл. 3. Формат:

CONVERT вводной-файл выводной-файл

Записи читаются из вводного файла последовательно и переписываются в выводной файл файловой организации, зависящей от указания ключей.

Команда COPY копирует файлы, сохраняя файловую организацию вводного файла. При копировании файла не переносится его код защиты. Формат:

COPY вводной-файл выводной-файл

где вводной-файл — спецификация одного или нескольких файлов, подлежащих копированию;
выводной-файл — спецификация файла, в который копируются вводные файлы. Необходимо обладать правом доступа по записи в каталог, куда помещается копия.

Если в спецификации выводного файла в качестве имени и типа заданы символы маскирования, имя и тип выводного файла будут такими же, как у вводного файла.

Если в каталоге уже существует файл с указанным именем и типом, то будет создан новый файл с версией на 1 больше существующего.

Если в спецификации выводного файла указан номер версии, то будет создан файл с этим номером версии. Если такой файл уже существует, то выполнение операции будет зависеть от наличия в командной строке ключа /REPLACE.

Если в командной строке указано несколько вводных файлов, а спецификация выводного файла задана символом маскирования, будут созданы копии указанных вводных файлов с такими же именами и типами.

Ключи:

/ALLOCATION:N — указывает, что N последовательных блоков резервируются для новой копии файла. Если не указана десятичная точка, N является восьмеричным числом;
/BLOCK_SIZE:N — определяет размер блока для магнитной ленты;
/ [NO] CONTIGUOUS — задает условие непрерывности выводного файла. Если ключ не указан, то непрерывными будут только копии тех файлов, которые были непрерывными;
/DATE:DD-MMM-YY— указывает, что копируются только файлы с указанной датой создания;
/EXCLUDE: спецификация-файла — определяет, что из операции копирования исключаются указанные файлы;
/NONEW_VERSION — запрещает создание новой версии при копировании файла;
/NOWARNINGS — подавляет вывод сообщений об ошибках;

/OVERLAY — указывает, что содержимое выводного файла перед копированием уничтожается, но идентификатор файла остается неизменным. Выводной файл должен уже существовать;

/OWN — изменяет код владельца файла при копировании в соответствии с именем выводного каталога;

/PRESERVE_DATA — указывает, что дата создания выводного файла должна быть такой же, как дата создания вводного файла. По умолчанию датой создания выводного файла является текущая дата;

/REPLACE — указывает, что если на выводном томе уже существует файл с таким же именем, расширением, версией, что и копируемый, то он удаляется и заменяется на копируемый файл;

/REWIND — вызывает перемотку ленты на начало перед копированием;

/SHARED — указывает, что другие пользователи могут иметь доступ к копируемому файлу;

/SINCE: DD-MMM-YY— определяет, что копируются файлы с датой создания, начиная с указанной;

/THROUGH: DD-MMM-YY — определяет, что копируются файлы с датой создания до указанной включительно;

/TODAY — копируются файлы с текущей датой создания.

Пример:

COPY/OWN TSKBLD.CMD [AB] BLDFIL.CMD

В данном примере файл из текущего каталога копируется в каталог (AB) с именем BLDFIL.CMD и с изменением UIC владельца.

Команда CREATE создает последовательный файл на устройстве с файловой структурой. После команды может сразу вводиться текст. Для закрытия файла вводится CTRL/Z. Формат:

CREATE спецификация-файла

Команда CREATE/DIRECTORY создает каталог (UFD) на томе с файловой структурой и заносит его имя в главный каталог. Формат:

CREATE/DIRECTORY [DDNN:] [каталог]

где DDNN: — спецификация устройства, на котором находится монтированный том. По умолчанию SY:;

[каталог] — имя создаваемого каталога, по умолчанию текущий. Каталог может иметь именованную или числовую форму.

Ключи:

/ALLOCATION:N — указывает число записей каталога (имен файлов), для которых необходимо зарезервировать пространство в файле каталога. По умолчанию N равно 32;

/LABEL: метка-тома — указанная метка тома сравнивается с настоящей меткой на томе, и в случае совпадения выполняется команда. Если метки не совпадают, команда отвергается;

/NOWARNINGS— подавляет вывод сообщения об ошибках;

/OWNER_UIC: [код-идентификации-пользователя] — указывает владельца каталога. UIC устанавливает статус защиты каталога;

/PROTECTION: (код)— определяет код защиты для файла каталога.

Команда DEALLOCATE освобождает личное устройство и делает его общедоступным. Непривилегированный пользователь может освобождать только свои устройства. Формат:

DEALLOCATE DDNN;

Ключи:

/ALL — освобождает все личные устройства, зарезервированные с терминала TI: ;

/DEVICE — не производит никаких действий;

/TERMINAL:TTNN: — позволяет привилегированному пользователю освободить устройство, зарезервированное с данного терминала.

Команда DEASSIGN отменяет логическое имя, назначенное командами ASSIGN или DEFINE. Она разрывает связь логических имен с именами физических, логических и псевдоустройств. Формат:

DEASSIGN логическое-имя

где логическое-имя — для систем, которые поддерживают расширенные логические имена, указывает назначение логического имени, которое необходимо удалить.

Логическое имя должно быть заключено в двойные кавычки в следующих случаях:

1) если оно оканчивается двоеточием (:) ;

2) если оно содержит символы, отличные от букв от A до Z, цифр от 0 до 9, знаков — двоеточие (:), подчеркивание (_) и знака $.

Для систем, которые не поддерживают расширенные логические имена, аргумент команды указывает логическое имя устройства, которое необходимо удалить.

Ключи:

/ALL — удаляет все назначения логических имен;
/GLOBAL — удаляет все глобальные (системные) назначения логических имен из системной таблицы логических имен. Ключ является привилегированным. С ним нельзя указывать ключ /TERMINAL;
/GROUP [:G] — удаляет групповое логическое имя из таблицы логических имен группы. Аргумент G указывает номер группы;
/LOCAL — удаляет локальные логические имена из таблицы логических имен пользователя;
/LOGIN — удаляет логические имена, установленные при регистрации;
/SYSTEM — то же, что и /GLOBAL;
/TERMINAL:TTNN: — удаляет логическое имя, назначенное для указанного терминала.

Команда DEASSIGN/QUEUE используется для разрыва связи очереди заданий с процессом обработки в подсистеме диспетчера очередей. Ф о р м а т :

DEASSIGN/QUEUE имя-очереди имя - процессора

Команда DEBUG переводит задачу в отладочный режим, устанавливая в PSW бит T. В ОСРВМ имеются отладчик ODT и отладчик DEBUG для программ, написанных на Фортран-77. Ф о р м а т :

DEBUG имя-задачи

где имя - задачи — имя отлаживаемой задачи. Если оно не указано, то используется имя терминала, с которого была выполнена команда.

Команда DEFINE приравнивает логическое имя текстовой строке или другому логическому имени. Назначения, выполненные командой DEFINE, удаляются командой DEASSIGN или переназначаются. Ф о р м а т :

DEFINE логическое-имя эквивалентное-имя

где логическое-имя — имя, назначаемое устройству или спецификации файла;
эквивалентное-имя — символьная строка, которая заменяет логическое имя.

К л ю ч и :

/FINAL — указывает, что трансляция логического имени заканчивается текущим значением эквивалентной строки;
/GLOBAL — указывает, что назначение будет помещено в системную таблицу логических имен. Ключи /GLOBAL и /SYSTEM являются синонимами;
/GROUP:G — делает назначение логического имени действительным для пользователей указанной группы. Если аргумент G не указан, по умолчанию принимается номер группы пользователя UIC по защите;
/LOCAL — определяет локальное назначение (действительное только для данного терминала). Ключ принимается по умолчанию. Отмена назначения не производится при размонтировании устройства;
/LOGIN — определяет логическое назначение как регистрационное. Ключ привилегированный. Такое логическое назначение действительно только на время пользовательской сессии;
/TERMINAL;TTNN: — указывает, что локальные назначения делаются для определенного терминала;
/TRANSLATION_ATTRIBUTES = TERMINAL — синоним ключа /FINAL. Используется для совместимости с МОСВП.

Команда DELET предназначена для удаления файлов. Ф о р м а т :

DELET спецификация-файла

Может быть указана спецификация одного или нескольких файлов, если спецификаций файлов несколько, их следует разделять запятыми.

К л ю ч и :

/DATE: DD-MMM-YY — указывает, что будут удаляться файлы с указанной датой создания;
/EXCLUDE:спецификация-файла — определяет файлы, исключаемые из операции удаления;
/LOG — распечатывает на TI: имена удаляемых файлов;
/QUERY — запрашивает разрешение на удаление каждого файла. В качестве ответа вводится один из следующих символов;
Y — удалить;
N — не удалять;
G — удалить все оставшиеся специфицированные файлы;
Q — не удалять файл и завершить работу команды;
Символ RETURN действует, как N, а CTRL/Z — как Q;
/SINCE: DD-MMM-YY — определяет, что должны быть удалены файлы, созданные начиная с указанной даты;
/THROUGH: DD-MMM-YY — определяет, что должны быть удалены файлы, созданные до указанной даты;
/TODAY — указывает, что будут удалены файлы с текущей датой создания;
/NOWARNINGS — подавляет выдачу сообщений об ошибках.

Команда DELETE/DIRECTORY удаляет каталог с тома, имеющего файловую структуру, и запись о нем из главного каталога MFD. Ф о р м а т :

DELETE/DIRECTORY DDNN: каталог

где DDNN: каталог — спецификация удаляемого каталога, содержащая хотя бы его имя.

Команда DELETE/ENTRY удаляет задание из очереди по номеру задания или файлы из указанного задания. Ф о р м а т :

DELETE/ENTRY:N
где N — номер задания в очереди.

К л ю ч :

/FILE_POSITION: N — указывает номер файла из последовательности файлов заданий на печать.

Команда DELETE/JOB удаляет задания по имени из указанной очереди. Ф о р м а т :

DELETE/JOB имя-очереди имя-задания

где имя-очереди — указывает на очередь, содержащую удаляемое задание;
имя-задания — имя удаляемого задания. Состоит из UIC пользователя, имени первого файла, в задании или имени, указанного ключом /NAME в командах PRINT или SUBMIT.

К л ю ч /FILE_POSITION: N аналогичен ключу предыдущей команды.

Команда DELETE/тип-процессора удаляет указанный процессор обработки из подсистемы диспетчера очередей по имени процессора или имени устройства. Ф о р м а т :

DELETE/тип-процессора имя-процессора

где тип-процессора — APPLICATION_PROCESSOR, BATCH_PROCESSOR, DEVICE, PRINTER или ROCESSOR;
имя-процессора — имя процессора обработки пакетных заданий или вывода на печать.

Команда DELETE/QUEUE удаляет очереди в подсистеме диспетчера очередей по имени. Команда привилегированная, ключ является обязательным. Ф о р м а т :

DELETE/QUEUE имя-очереди/ERASE

Команда DIFFERENCES сравнивает два символьных файла построчно для выявления различий и создает протокол различий. Ф о р м а т :

DIFFERENCES вводной-файл-1 вводной-файл-2

где вводной-файл-1 и вводной-файл-2 — спецификации сравниваемых файлов.

К л ю ч и :

/CHANGE BAR [:N] — определяет, что в качестве выводного файла выдается листинг вводного файла 2 с пометками об изменениях в файле 2, не имеющих места в файле 1. Аргумент N — это код символа, используемого для отметки несовпадений. По умолчанию N равно 041 (символ !);
/IGNORE: (аргумент [, ...]) — указывает элементы текста, которые игнорируются при сравнении. Возможны следующие значения аргумента:

BLANCK LINES — пустые строки при сравнении не учитывать;
COMMENTS — комментарии при сравнении не учитывать;
FORM FEED — игнорировать строки, начинающиеся символом FF;
SPACING — последовательность пробелов и табуляций рассматривать как один пробел;
TRAILING BLANCKS — пробелы, следующие за последним символом строки, игнорировать;
/LINES:N — указывает количество идущих подряд идентичных записей, служащих признаком окончания секции несравнения. По умолчанию N равно 3;
/[NO]NUMBERS — указывает, будут ли пронумерованы строки в выводном файле. По умолчанию они нумеруются, начиная с единицы;
/OUTPUT :спецификация-файла — указывает, что выводной листинг будет помещен в указанный файл. По умолчанию вывод осуществляется на терминал TI:;
/SPL [:строка-символов] — указывает, что выводной листинг должен иметь формат косвенного командного файла SPL, который позволит сделать вводной файл 1 идентичным вводному файлу 2.

Команда DIRECTORY распечатывает информацию о файлах в каталогах. Ф о р м а т :

DIRECTORY [спецификация-файла]

где спецификация-файла — спецификация файла или файлов, о которых запрашивается информация. При отсутствии параметра выводится содержимое каталога по умолчанию.

К л ю ч и :

/ATTRIBUTES — вызывает вывод атрибутов файла: имя, расширение, версия, дата и время создания, организация файла, код защиты, количество блоков, зарезервированных для файла, дата последней модификации файла, формат и размер записей, главный и дополнительный ключи для индексного файла, размер бакета для индексных и относительных файлов;
/BRIEF — распечатывает краткую информацию о файле, содержащую только имя, тип и номер версии файла;

/DATE: DD-MMM-YY — указывает, что должна выводиться информация о файлах с указанной датой создания;
/EXCLUDE: спецификация файла — указывает файлы, исключаемые из операции;
/FREE [DDNN:] — распечатывает количество свободных блоков и число свободных заголовков файлов на томе DDNN: (по умолчанию SY:);
/FULL — распечатывает полную информацию о файле в каталоге (см. разд. 7);
/NOWARNINGS — подавляет вывод сообщений об ошибках;
/OUTPUT: спецификация файла — указывает, что информация помещается в указанный файл;
/PRINTER — указывает, что вывод направляется на LP:;
/REWIND — вызывает перемотку ленты на начало перед операцией;
/SINCE: DD-MMM-YY — указывает, что информация будет выводиться о файлах с датой создания, начиная с указанной;
/SUMMARY — распечатывает только общее число зарезервированных и используемых блоков для указанных файлов. Если в команде опущены спецификации файлов, то распечатывается общее число зарезервированных и использованных блоков для каталога, установленного по умолчанию;
/THROUGH: DD-MMM-YY — указывает, что будет выводиться информация о файлах, созданных до указанной даты;
/TODAY — указывает, что будет выводиться информация о файлах с текущей датой создания.

Команда DISMOUNT служит для демонтирования томов. Указанный том отмечается как логически отсоединенный от файловой системы. После того как все открытые файлы на нем будут закрыты, том будет демонтирован. Сообщение о завершении операции выдается на консоль. Ф о р м а т :

DISMOUNT DDNN: [метка]
где DDNN: — это устройство, на котором монтирован том; метка — метка монтированного тома.

К л ю ч и :

/ALL — указывает, что должны размонтироваться все устройства, монтированные с данного терминала;
/PUBLIC — вызывает размонтирование тома для всех пользователей, монтировавших его;
/SAVE — указывает, что диск остается доступным для привилегированных задач. Применим для устройств DM: и DU:;
/SYSTEM — то же, что и /PUBLIC;
/TERMINAL: TTNN: — позволяет привилегированным пользователям размонтировать том, монтированный с указанного терминала;
/NO_UNLOAD — указывает, будет ли устройство отключено после демонтирования.

П р и м е р :

```
DISMOUNT/PUBLIC DUO:
DMO_ _TT1: DISMOUNT FROM DUO:
DMO_ _TT2: DISMOUNT FROM DUO:
DMO_ _TT5: DISMOUNT FROM DUO:

***FINAL DISMOUNT***
```

В данном примере привилегированный пользователь размонтировал том DUO: для всех пользователей, которые его монтировали.

Команда EDIT [/EDT] вызывает текстовый редактор EDT. Ф о р м а т :

EDIT [/EDT] – спецификация-файла где спецификация-файла — файл, предназначенный для редактирования.

К л ю ч и :

/[NO] COMMAND [:спецификация-файла] — указывает, будет ли перед редактированием выполняться командный файл EDT, изменяющий значения по умолчанию;
/[NO] CREATE — указывает, будет ли создаваться новый файл, если специализированный не найден;
/[NO] JOURNAL [:спецификация-файла] — указывает, будет ли создаваться журнальный фонд, содержащий команды редактора и вводимый текст для текущего сеанса редактирования. По умолчанию создается журнальный файл с таким же именем, как у вводного файла и типом .JOU;
/[NO] OUTPUT [:спецификация-файла] — указывает, будет ли создан выводной файл. По умолчанию создается выводной файл с таким же именем, как и вводной, и с версией на единицу больше;
/[NO] READONLY — позволяет просматривать файл без редактирования;
/[NO] RECOVER — определяет, будет ли EDT читать команды из журнального файла до сеанса редактирования при повторной обработке файла.

Команда EDIT/SLP вызывает пакетный редактор SLP. Ф о р м а т :

EDIT/SLP спецификация-файла

К л ю ч и :

/[NO] AUDIT [:аргумент] — указывает, будет ли выводной файл содержать индикатор автоматического слежения и дополнительно позволяет установить положение и размер индикатора. Если указаны оба аргумента, их следует заключать в круглые скобки и разделять запятыми.

Возможны следующие значения аргумента:

POSITION:N — определяет начальную позицию индикатора. N может принимать значения от 0 до 132. По умолчанию

равно 80;

SIZE: N — определяет размер индикатора. N может принимать значения от 0 до 14. По умолчанию N равно 80;

/[NO] CHECKSUM [:аргумент] — управляет вычислением контрольной суммы для команд SLP. Если аргумент опущен, то SLP вычисляет контрольную сумму и выводит на терминал. Если аргумент указан, то SLP сравнивает вычисленную контрольную сумму с аргументом и при несравнении выводит сообщение на терминал;

/[NO] LIST [:спецификация-файла] — указывает, будет ли создан листинг под указанным именем;

/[NO] OUTPUT [:спецификация-файла] — используется для изменения имени выводного файла;

/[NO] REPORT — управляет выводом на терминал сообщений об усечении строк;

/[NO] TAB — управляет подстановкой пробелов и знаков табуляции в конце каждой записи;

/[NO] TRUNCATE [:N] — управляет усечением каждой записи вводного файла перед коррекцией. N должен быть не больше 132, указывает, с какой позиции усекается текст.

Команда EDIT/EDI вызывает текстовый редактор EDI.

К л ю ч :

/EDI.

Команда EDIT/редактор вызывает редактор, определенный пользователем.

К л ю ч :

/USING: YYY указывает, что будет использоваться редактор, установленный пользователем под именем YYY.

Команда FIX служит для фиксации в памяти установленной задачи или общей области, которые становятся резидентными в памяти. Задачи, фиксируемые в памяти, как правило, должны быть, повторно входимыми. Ф о р м а т :

FIX имя-задачи

К л ю ч и :

/READONLY_SEGMENT — позволяет фиксировать в памяти сегмент с атрибутом RO (только чтение) многопользовательской задачи; /REGION — позволяет фиксировать в памяти общую область.

Команда HELP выводит на терминал справочную информацию о системе. Ф о р м а т :

HELP [% параметр-1, ..., параметр-N]

где параметр-N — команда или ключ, о которых требуется распечатать информацию. Эта же информация выводится при вводе символа «?» в ответ на подсказку.

К л ю ч и :

/CLI:имя:интерпретатора — указывает, что будет выводиться информация для указанного интерпретатора команд из файла LB: [1,2] имя CLI.HLP;

/DCL — указывает, что будет выводиться информация о командах DCL (по умолчанию);

/FILE: спецификация-файла — в этом файле содержится дополнительная информация;

/имя-файла — указывает, что информация находится в файле LB: [1,2] имя-файла .HLP;

/GROUP — определяет, что информация находится в каталоге с текущим номером группы и номером пользователя, равным 1;

/LOCAL (или %) — определяет, что вспомогательная информация находится в файле HELP.HLP на устройстве и в каталоге, установленном по умолчанию;

/MCR — указывает, что будет выводиться информация о командах MCR, хранящаяся в файле LB: [1,2] MCR.HLP. Этот ключ считается заданным по умолчанию для терминалов, установленных в режим MCR;

/OUTPUT: спецификация-файла — указывает спецификацию файла, используемого для вывода полученной информации.

Команда HOLD/ENTRY задерживает в очереди задание с указанным номером (см. разд. 6). Ф о р м а т :

HOLD/ENTRY: N

где N — номер элемента в очереди.

Команда HOLD/JOB задерживает в указанной очереди задание с указанным именем. Ф о р м а т :

HOLD/JOB имя-очереди имя-задания

Команда INITIALIZE создает файловую структуру ОСРВМ на томе. На дисках команда **INTIALIZE** разрушает существующие файлы, записывает пустой блок начальной загрузки, собственный блок тома и создает структуру каталога. На лентах создается стандартная метка тома и пустой файл. Ф о р м а т :

INITIALIZE DDNN: метка-тома

где DDNN: — спецификация устройства, на котором установлен том; метка-тома — присваивается тому при инициализации. Для дисков до 12 символов, для лент — до 6.

К л ю ч и :

/ACCESSED: N — указывает число каталогов (UFD), которые будут одновременно находиться в системной динамической памяти и соответственно за ними не нужно обращаться к диску. Аргумент должен быть не более 127, по умолчанию — 3;

/BAD_BLOCK: аргумент — определяет расположение файла описателя
дефектных блоков на диске. Возможны следующие значения аргумента:

AUTOMATIC — читает файл описателя дефектных блоков, созданный командой ANALYZE/MEDIA, и автоматически по нему создает файл BADBLK.SYS. Это значение принимается по умолчанию;

(AUTOMATIC, MANUAL) — читает файл описателя дефектных блоков и после этого принимает список дефектных блоков, введенных с терминала;

MANUAL — список дефектных блоков запрашивается с терминала;

NOAUTOMATIC — игнорирует файл описателя дефектных блоков и не выполняет обработку дефектных блоков;

OVERRIDE — игнорирует файл описателя дефектных блоков на последней дорожке, читает файл, созданный командой ANALYZE/MEDIA/OVERRIDE в последнем хорошем блоке перед последней дорожкой, и автоматически создает файл BADBLK.SYS. Этот аргумент, как и следующий, используется только для дисков DM: и DR:;

(OVERRIDE, MANUAL) — игнорирует файл описателя дефектных блоков на последней дорожке, читает файл описателя, созданный командой ANALYZE/MEDIA/OVERRIDE в последнем недефектном блоке перед последней дорожкой, автоматически создает файл BADBLK. SYS, запрашивает список дефектных блоков с терминала;

/DENSITY: аргумент — определяет плотность записи на томе.

Аргумент может принимать значения:

для лент — 800 или 1600, что соответствует плотности записи 32 бита/мм или 64 соответственно;

для гибких дисков LOW — одинарная плотность записи, HIGH — двойная;

/EXTENSION: N — определяет число блоков, на которое расширяется файл, когда отведенное ему пространство исчерпано. По умолчанию N=5;

/FILE_PROTECTION: (код) — определяет код защиты по умолчанию для всех файлов тома;

/HEADERS: N — указывает число заголовков файлов, для которых первоначально зарезервировано место в индексном файле. Пять системных- файлов в это число не включаются. Значение по умолчанию выбирается в зависимости от емкости тома, минимально — 16. В зависимости от размера диска сам индексный файл может иметь один, два или три заголовка.

Имеются формулы для расчета числа заголовков файлов в зависимости от типа индексного файла. Для индексного файла с одним заголовком N = MAX/2, где N — число первоначально размещенных заголовков, MAX — максимальное число файлов, указанное ключом /MAXIMUM_FILES. Если N<100, то выделяется 16 заголовков.

Для индексного файла с двумя заголовками N = MAX/2, или N не меньше 25593.

Для индексного файла с тремя заголовками N=MAX/2, где N не меньше 51699:

/INDEX: аргумент — определяет место на томе, на котором находится индексный файл, каталог (UFD) и файл распределения памяти. Возможны следующие значения аргумента:

BEGINING — файлы размещаются в начале тома;

MIDDLE — файлы размещаются в середине тома;

END — файлы размещаются в конце тома;

N — файлы размещаются, начиная с указанного логического блока;

/LABLE: VOLUME_ACCESSIBILITY: «символ» — определяет код защиты доступа для тома на ленте. Эта команда помещает указанный символ в пустое поле доступа метки VOLI:

/MAXIMUM_FILES: N — определяет максимальное число файлов, которое можно разместить на томе.

$$N = \frac{\left(X\left(\left(X + 4095\right)/4096\right) + 9\right) * 127}{258},$$

где X — размер диска в блоках;

/OWNER: [G, M] — определяет владельца тома. Значение ключа используется для проверки тома и кода защиты файла;

/PROFESSIONAL — определяет некоторые умолчания при инициализации тома на гибких дисках RX50. Системные файлы при указании этого ключа
помещаются в начале тома в каталоге по умолчанию [200, 200]. Если метка тома не указана, то она генерируется из первых 12 символов текущей даты и времени. Кроме того, устанавливаются следующие значения по умолчанию:

/FILE_PROTECTION — доступ всем классам пользователей RWED;

/MAXIMUM_FILE = 200;

/PROTECTION: (код) — определяет код защиты по умолчанию для новых файлов, создаваемых на томе;

/[NO] SHOW — указывает, требуется ли распечатка всех значений параметров, с которыми том инициализируется. По умолчанию — /NOSHOW;

/WINDOWS: N — определяет размер блока указателей извлечения файла для окон файла. Окно состоит из размера указателей извлечения файла, помещаемых в памяти, когда файл открывается. По умолчанию N = 7.

Команда INITIALIZE/PROCESSOR создает, именует и запускает процессор вывода на печать или процессор пакетной обработки подсистемы диспетчера очередей. Ф о р м а т :

INITIALIZE/тип-процессора имя-процессора

Подробное описание этой и следующей команд приведено в разд. С.

Команда INITIALIZE/QUEUE создает, именует и запускает очередь в подсистеме диспетчера очередей. Ф о р м а т :

INITIALIZE/QUEUE имя-очереди

Команда INITIALIZE/UPDATE изменяет параметры тома на диске, установленные командой INITIALIZE, не изменяя данные на томе. Явно не указанные параметры остаются без изменений. Ф о р м а т :

.INITIALIZE/UPDATE DDNN: метка-тома

где DDNN: — спецификация устройства.

Значение ключей /ACCESSED, /DENSITY, /EXTENSION: N, /OWNER, /PROFESSIONAL, /PROTECTION, /SHOW, /WINDOWS совпадает с ключами команды INITIALIZE.

К л ю ч и :

/FILE_PROTECTION: (код) — устанавливает код защиты для новых файлов, создаваемых на томе;

/LABEL: новая-метка-тома — позволяет изменить метку тома, которая может содержать до 12 символов;

/MAXIMUM_FILES: N — указывает максимальное количество файлов, размещаемых на томе. Значение N должно быть больше, чем текущее максимальное значение.

Команда INSTALL устанавливает задачу, включая и список установленных задач (STD), тем самым делая ее известной системе. Ключи команды INSTALL имеют приоритет перед аналогичными ключами ТКВ. Ф о р м а т :

INSTALL [$] спецификация-файла-образа-задачи

Наличие символа [$] указывает, что файл следует искать в системном каталоге.

К л ю ч и :

/[NO] CHECKPOINT — определяет признак выгружаемости задачи;

/COMMAND: «команда» — используется для передачи команды устанавливаемой задаче. Размер команды не должен превышать 40 символов;

/EXTENSION: N — расширяет адресное пространство задачи на N слов;

/NO INTERPRETER — определяет, является ли устанавливаемая задача интерпретатором команд (CLI). CLI должен быть установлен с этим ключом до выдачи команды SET TERMINAL/CLI;

/MULTIUSER PARTITION имя-раздела — определяет раздел, в котором будет установлена часть многопользовательской задачи с кодом доступа RO (только чтение);

/PARTITION: имя-раздела — определяет раздел, в котором будет установлена задача;

/[NO] POSTMORTEM — определяет, необходим ли запуск программы распечатки памяти при аварийном завершении задачи;

/PRIORITY: N — указывает приоритет выполнения задачи. Значение по умолчанию назначается во время построения. N может принимать значения от 0 до 250;

/READONLY_COMMON — указывает, что общая область устанавливается как область с атрибутом RO (только чтение);

/[NO] RESIDENT_HEADER — указывает, будет или нет копия заголовка задачи помещаться в системную динамическую область (пул). Если указан ключ /RESIDENT_HEADER, задача будет установлена с резидентным заголовком;

/[NO] SLAVE — определяет, будет ли задача подчиненной. Если указан этот ключ, то при приеме задачей данных, посылаемых другой задачей, ей передается в качестве устройства терминал TI:, с которого загружена задача, посылающая данные;

/TASK_NAME: имя — определяет имя задачи, по которому она вызывается;

/TRANSLATION_ROUTINE: N — загружает программу преобразования символов и драйвер терминала, осуществляет трансляцию между различными наборами символов. Преобразование символов в драйвере терминала позволяет работать на терминалах с нестандартным набором символов. К терминальному драйверу присоединяется вспомогательный (ACD), который управляет программой преобразования символов.

При использовании данного ключа полный формат команды:

INSTALL/TRANSLATION_ROUTINE N спецификация файла [логическое-имя]

Аргумент N идентифицирует вспомогательный драйвер, который управляет программой преобразования символов, содержащейся в указанном файле. Необязательный параметр «логическое-имя» присваивается ACD;

/UIC: [G, M] — указывает UIC по умолчанию для задачи;

/[NO] WRITEBACK — определяет, будет ли общая область записываться в исходный файл образа задачи при выгрузке или удалении. По умолчанию /NOWRITEBACK, т. е. общая область будет выгружаться в системный файл.

Команда LIBRARY создает и поддерживает библиотечные файлы, работа с которыми подробно описана в разд. 7, Ф о р м а т :

LIBRARY/операция

или

LIBRARY спецификация-командного-файла

К л ю ч и . Каждая из операций соответствует определенному ключу библиотекаря (LBR):

Операция	Ключ LBR
COMPRESS	/CO
CREATE	/CR
DELETE	/DE
EXTRACT	/EX
INSERT	/IN
LIST	/LI
REMOVE	/DG
REPLACE	/RP

Команда LINK вызывает построитель задач ТКВ, который компонует объектные модули, созданные трансляторами, в образ задачи. Ф о р м а т :

LINK спецификация-файла

где спецификация-файла или файлов определяет файлы типа OBJ или OLB, используемые для компоновки.
Информация о ключах и параметрах команды приведена в разд. 5 при описании ТКВ.

Команда LOGIN разрешает доступ в систему с терминала аналогично команде MCR HELLO. Когда пользователь регистрируется в системе, для него устанавливаются некоторые характеристики терминальной сессии (привилегии, устройство по умолчанию и т. д.). Ф о р м а т :

LOGIN идентификатор-пользователя/пароль

где идентификатор-пользователя — указывает пользователя, в настоящий момент регистрирующегося в системе. Имя пользователя ранее было внесено в файл учетной информации. При этом допускаются четыре формы представления кода идентификации пользователя:

[G,M] G,M [G/M] G/M

Если при регистрации пользователя в системе используются имя или UIC в формах, содержащих запятую, LB: [1,2] LOGIN.TXT выводится на терминал;
пароль — сочетание символов RADIX-50 числом не более 39, установленных для данного пользователя.

Команда LOGOUT регистрирует выход пользователя их системы и делает терминал доступным для других пользователей. LOGOUT также аварийно завершает все свои непривилегированные задачи, запущенные с терминала, демонтирует все личные тома и отсоединяет все личные устройства, присоединенные с терминала, включая те, которые были зарезервированы командой MOUNT/NOSHAREABLE. Ф о р м а т :

LOGOUT

К л ю ч :

/[NO] HOLD используется для терминалов, подключенных через линии связи. По ключу /HOLD осуществляется выход из системы, но линия не отсоединяется.

Команда MCR, адресованная DCL, обеспечивает выполнение одной команды MCR, которая задана в командной строке. Выдается с терминала, для которого интерпретатором командной строки является DCL. Ф о р м а т :

MCR команда

Команда MOUNT (MOU) объявляет том доступным. Ф о р м а т :

MOUNT DDNN:метка-тома [/ключ(и)]
MOUNT DDNN: [,DDNN: ...] идентификатор-набора-файлов

Первая командная строка относится к дискам, вторая командная строка относится к лентам. Идентификатор набора может быть не более 6 символов.
При монтировании магнитной ленты можно задать список устройств, заключая его в круглые скобки.

К л ю ч и :

/DEFAULT: аргумент — устанавливает значение по умолчанию при размонтировании тома. Аргумент может принимать следующие значения:
SAVE — привилегированный, указывает, что том на магнитной ленте должен разгружаться и будет доступен

привилегированным задачам;

[NO] UNLOAD — определяет, будет ли носитель разгружен после размонтирования. UNLOAD (разгружать) — значение по умолчанию для всех устройств с файловой структурой. NOUNLOAD (не разгружать) — для носителей, монтированных с ключом /FOREIGN. Ключ применяется для магнитных лент и дисков типа DM: и DU:;

/FILE_PROTECTION: (код) — определяет код защиты вновь создаваемого файла по умолчанию;

/FOREIGN — монтируемый том не содержит файловой структуры OCPBM;

/OVERRIDE — разрешает привилегированному пользователю монтировать том без указания метки;

/PARAMETERS: «параметры-пользователя» — позволяет передать до 40 символов задаче ACP, которая обрабатывает том, не содержащий файловую структуру OCPBM (см. также ключ /FOREIGN);

/PROCESSOR: аргумент — позволяет привилегированному пользователю задать в качестве аргумента имя ACP, который будет обрабатывать том, не содержащий файловую структуру OCPBM (см. также ключи /FOREIGN и /PARAMETERS). Кроме того, ключ применяется для создания копии стандартного ACP, который будет обрабатывать монтируемый том (аргумент — UNIQUE). После размонтирования копия удаляется;

/PROTECTION: (код) — определяет защиту тома для дисков с файловой структурой OCPBM;

/PUBLIC — привилегированный ключ, который указывает, что том доступен всем пользователям с соответствующими привилегиями доступа;

/[NO] SHAREABLE — определяет, что к тому разрешен или запрещен (/SHAREABLE или /NOSHAREABLE соответственно) раздельный доступ.

Том может многократно монтироваться разными пользователями. Обращение регламентируется, кроме того, привилегиями доступа;

/[NO] SHOW — разрешает (/SHOW) или запрещает (/NOSHOW) вывод на терминал информации о монтируемом томе;

/[NO] WAIT — определяет, что требуется вмешательство оператора (/WAIT) при попытке монтирования устройства, находящегося в состоянии «не готов». Если задано /NOWAIT и устройство в состоянии «не готов», оператор консольного терминала не извещается о состоянии устройства;

/[NO] WRITE — разрешает (/WRITE) операцию записи на том в соответствии с привилегиями пользователя. /NOWRITE запрещает операцию записи.

Специальные ключи для монтирования дисков:

/ACCESSED: N — определяет число одновременно доступных каталогов, под которые резервируется память в системном пуле, где N должен быть не меньше 1 и не больше 127;

/[NO] CAGHE:(аргумент, аргумент ...) — разрешает кэширование данных для диска или изменяет его характеристики. Аргументы:

CREATE [= область] [: [раздел] [: [размер]]] — область (задается ее имя) создается в главном разделе (указывается имя раздела) и размер области в дисковых блоках. По умолчанию — имя CACHE, раздел — GEN, размер — 100 блоков;

[NO]DIRECTORY [= размер-экстента] — разрешает или запрещает (NODIRECTORY) кэширование операций ввода-вывода каталога. Размер экстента задается в дисковых блоках, по умолчанию — 1;

[NO] LOGICAL [= размер-экстента] — разрешает или запрещает кэширование операций ввода-вывода логического блока (физический уровень ввода-вывода). По умолчанию — LOGICAL. Для аргумента LOGICAL размер по умолчанию равен 1 (размер экстента в блоках);

[NO] OVERLAY [= размер-экстента]— разрешает или запрещает кэширование операций ввода-вывода для загрузки сегментов задач с перекрытиями. По умолчанию OVERLAY=4. Размер экстента — в блоках;

[NO] VIRTUAL [= размер-экстента]— разрешает или запрещает кэширование операций ввода-вывода виртуальных блоков. По умолчанию размер экстента равен 5 (в блоках);

[NO] READ_AHEAD [= размер-экстента] — разрешает или запрещает чтение следующего экстента файла перед тем, как поступит явный запрос чтения. По умолчанию — NOREAD_AHEAD. По умолчанию для READ_AHEAD размер равен 5 (в блоках).

Для всех аргументов ключа /CACHE размер экстента в блоках не может превышать 15;

/EXTENSION: N — определяет число блоков, на которое будет увеличен файл, когда ранее выделенное количество блоков будет исчерпано. N должен быть не менее 1 и не более 127. По умолчанию действует N, установленный при инициализации тома;

/OWNER: [G, M] — определяет UIC владельца тома;

/UNBLOCK — указывает, что индексный файл доступен для чтения и записи. По умолчанию — только для чтения;

/WINDOW: { N | FULL | USER.N | INDEX: N | USER: N, INDEX: N }

где N — определяет число указателей извлечения файла, которые резервируются при открытии файла. N принимает значения от 1 до 127;

FULL — резервируется максимально возможное количество указателей извлечения файла;

USER: N — определяет число указателей извлечения для файлов пользователя;

INDEX: N — для индексного файла.

С п е ц и а л ь н ы е к л ю ч и для монтирования магнитных лент:

/BLOCK_SIZE: N —- определяет размер блока в байтах для магнитной ленты без меток или со стандартными метками, но без HDR2, N принимает значения от 1 до 127;

/CARRIGE_CONTROL: { FORTRAN | LIST | NOME } — определяет тип записи для магнитных лент без меток.

Аргументы соответствуют ключевым словам Фортрана для операции открытия файла:

/DENSITY: { 800 | 1600 } — задает плотность записи для магнитной ленты: 800 — обычная, 1600 — двойная;

/[NO] HDR3 — определяет, требуется ли запись метки HDR3 при создании нового файла. По умолчанию /HDR3;

/[NO] LABEL — определяет формат тома на магнитной ленте. /LABEL со стандартными метками, /NOLABEL — без меток;

/OVERRIDE:аргумент — позволяет привилегированному пользователю монтировать том, отвергая проверки, которые задаются аргументом. Аргумент имеет значения:

ACCESSEBILITY — отвергает проверку кода защиты для лент стандартного формата;
EXPIRATION ___ DATE — отменяет проверку срока хранения тома;
IDENTIFICATION — монтирование магнитной ленты без указания идентификатора данных;

/RECORD_SIZE: N — определяет размер записи в байтах для магнитной ленты без меток. N принимает значения от 1 до размера блока на томе;

/TRANSLATE: { EBCDIC | NONE | UT1, UT2, UT3 }

где EBCDIC — перекодировка из ДКОИ в КОИ при чтении и из КОИ и ДКОИ при записи;
NONE — без перекодировки;
UT1, UT2, UT3 — перекодировка с помощью пользовательских таблиц;

/VOLUME_IDENTIFICATION: (идентификатор 1 ...) — определяет идентификаторы томов для магнитных лент со стандартными метками. Применяется, если идентификатор набора файлов не совпадает с идентификатором тома первой ленты.

Команда PRINT ставит в очередь файлы для вывода на печатающее или на заданное устройство. Тип файла вывода по умолчанию — LST. Ф о р м а т :

PRINTспецификация-файла [,спецификация-файла...] [/ключ (и)]

К л ю ч и :

/AFTER: { DD-MMM-YY HH:MM:SS | MM-DD-YY HH:MM:SS | TOMORROW } — задает дату и время, до которого выполнение задания должно быть отложено. TOMORROW — отложить до следующего дня;

/COPIES:N — указывает число копий листинга выводного файла;

/[NO]DELET — определяет, будет ли указан выводной файл после распечатки. По умолчанию для файлов с типом LST — /NODELET (не удаляется), с типом DMP —/DELET;

/DEVICE: DDNN: — направляет ввод файла на заданное устройство;

/FORMS: N — задает тип формата, обеспечиваемый выводным устройством при печати. По умолчанию N = 0;

/[NO] FLAG_PAGE — определяет, будет ли добавляться флаговая страница к каждому файлу в задании. Флаговая страница будет предшествовать заданию, если задан /NOFLAG_PAGE, но файлы будут распечатаны без нее;

/[NO] HOLD — определяет задержку распечатки до выдачи команды RELEASE (/HOLD); по умолчанию /NOHOLD;

/JOB_COUNT: N — задает число копий распечатки задания. По умолчанию N=1;

/[NO] JOB_PAGE — требует вывода файлов флаговой страницы перед распечаткой задания (JOB_PAGE) или запрещает его; по умолчанию /JOB_PAGE;

/LENGHT: N — определяет количество строк на странице; принимает значения от 0 до 255. Перевод страницы осуществляется, если обнаружен символ <FF>, достигнут конец физической страницы или распечатаны N строк;

/[NO] LOVERCASE — определяет печать символов нижнего регистра. Если задано /LOWERCASE, то файл печати посылается на устройство, печатающее символы нижнего регистра; по умолчанию — /NOLOWERCASE;

/NAME : имя-задания — позволяет задать до 9 символов, которые идентифицируют задание на флаговой странице;

/PAGE_COUNT: N — устанавливает ограничения числа страниц при выдаче на печать; по умолчанию — нет ограничений;

/PRIORITY: N — приоритет задания в очереди на печать. Значения N от 0 до 150 — для непривилегированных пользователей, значения N от 0 до 250 — для привилегированных;

/QUEUE: имя очереди — определяет имя очереди заданий на распечатку; по умолчанию — PRINT;

/[NO]RESTART — задает повторный запуск сначала после остановки (/RESTART); по умолчанию —/NORESTART;

/[NO] UPPERCASE — указывает, будет ли задание распечатываться на устройстве только с символами верхнего регистра (UPPERCASE) или нижнего и верхнего (/NOUPPERCASE).

Команда PURGE удаляет все версии файлов, кроме последней, или указанного числа последних версий. Ф о р м а т :

PURGE спецификация-файла [/ключ(и)]

Спецификация файла указывается без указания версии.

К л ю ч и :

/DATE: DD-MMM-YY — задает дату создания файлов, которые следует удалить;
/EXCLUDE: спецификация-файла — задает файл (файлы), которые не должны удаляться;
/KEEP: N — задает N последних версий, которые должны быть сохранены, по умолчанию — /KEEP: 1;
/[NO]LOG — требует (/LOG) или запрещает (/NOLOG) распечатку имен удаляемых файлов на терминал;
/NOWARNINGS — запрещает выдачу сообщений об ошибках;
/SINCE: DD-MMM-YY — команда применяется к файлам с датой создания, начиная с указанной;
/THROUGH: DD-MMM-YY — команда применяется к файлам с датой создания до указанной в ключе даты;
/TODAY — команда применяется к файлам, созданным за текущий день.

Команда RELEASE освобождает задание, определенное номером элемента N, которое стоит в очереди задержанных заданий. Ф о р м а т ы :

RELEASE/ENTRY: N

RELEASE/JOB — имя-очереди имя-задания

Команда REMOVE удаляет задачу из списка установленных задач, разделяемую область или вспомогательный управляющий драйвер (/TRANSLATION_ ROUTINE : N) — из оперативной памяти (/REGION). Команда привилегированная. Ф о р м а т :

REMOVE имя задачи [{ /REGION | /TRANSLATION_ROUTINE: N }]

Команда RENAME переименовывает файл (файлы), изменяет тип файла, версию. Ф о р м а т :

RENAME вводной-файл выводной-файл [/ключ(и)]

К л ю ч и :

/DATE:DD-MMM-YY
/EXCLUDE:спецификация-файла
/NOWARNINGS
/SINCE:DD-MMM-YY
/THROUGH: DD-MMM-YY
/TODAY

Ключи описаны при рассмотрении программы PIP (см разд. 7).

Команда REQUEST выдает сообщение длиной до 80 байт на устройство CO:. Ф о р м а т :

REQUESTтекст-сообщения

Команда RUN предназначена для запуска неустановленной задачи, установленной задачи, установленной задачи в указанное время с требуемым интервалом перезапуска. Ф о р м а т ы :

RUN [$] спецификация-файла [/ключ(и)] RUN имя-задачи [/ключ(и)]

Первая командная строка для неустановленных задач, вторая — для установленных задач.

К л ю ч и д л я п е р в о г о ф о р м а т а :

/[NO]CHECKPOINT
/COMMAND: «командная-строка»
/EXTENSION: N
/[NO]IO_PAGE
/PAPTITION:имя-раздела
/[NO] POSTMORTEM
/PRIORITY: N
/READ_PARTITION: имя-раздела
/[NO] SLAVE /STATUS:аргумент
/TASK_NAME: имя-задачи
/TIME_LIMIT: N[U]
/UIC: [G,M]

К л ю ч и д л я в т о р о г о ф о р м а т а :

/DELAY:N(T,S,M,H)

/INTERVAL;N (T,S,M,H)
/SCHEDULE:HH:MM:SS
/STATUS
/SYNCHRONIZE: (T,M,S,H)
/UIC: [G,M]

Все ключи рассмотрены при описании команды RUN программы связи с оператором (MCR) [1].

Команда SET [DAY]TIME устанавливает время и/или дату в системе. Ф о р м а т :

SET [DAY] TIME [дата] [время]

Дата задается в одном на двух форматов:

1. DD-MMM-YY
2. MM-DD-YY

Время задается следующим образом:

[HH:MM:SS]

Команда SET DEBUG позволяет установить или запретить режим работы, при котором на экран выводятся команды MCR [1]. Ф о р м а т :

SET [NO] DEBUG [{ /NO EXECUTE } { /FULL }]

К л ю ч и :

/[NO] EXECUTE — определяет, будет ли выполнена команда DCL после ее преобразования в команду MCR;

/FULL — требует дополнительного вывода значений логических символов, используемых DCL при трансляции.

Команда SET DEFAULT позволяет установить по умолчанию устройство и каталог. Если команда задана без аргументов, то восстанавливаются значения по умолчанию, которые были установлены при регистрации в системе. Ф о р м а т :

SET DEFAULT [DDNN:] [каталог] [/ключ]

где DDNN: — спецификация устройства по умолчанию;
каталог — имя каталога по умолчанию. Имя не может превышать 9 символов и состоит из букв от A до Z и цифр от 0 до 9 [G,M] — формат цифрового каталога;
ключ /[NO] NAMED_DIRECTORY — определяет форму, в которой задается каталог. /NAMED-DIRECTORY — именованный каталог, /NONA-MED_DIRECTORY — цифровая форма задания каталога.

Команда SET DEVICE устанавливает характеристику устройства DDNN:. Команда привилегированная. Непривилегированный пользователь может изменять параметры только собственного терминала. Ф о р м а т :

SET DEVICE [:DDNN:] [/ключ]

Если устройство и ключ не указаны, они вводятся в ответ на соответствующую подсказку системы.

К л ю ч и :

/[NO] CHECKPOINT_FILE:N — разрешает или запрещает использование
файла выгрузки на томе, заданном параметром DDNN:. Том должен иметь файловую структуру OCPBM. N определяет размер файла выгрузки [0,0] CORIGM.SYS в блоках (десятичное число);
/[NO] LOWERCASE — управляет преобразованием букв нижнего регистра в буквы верхнего регистра при выводе на терминал и АЦПУ; /NOLOWERCASE— по умолчанию;
/[NO] PUBLIC — определяет доступность устройства всем пользователям (/PUBLIC). При этом устройство монтируется; по умолчанию /NOPUBLIC;
/[NO] SYSTEM — синоним ключа /[NO] PUBLIC;
/WIDTH:N — задает размер буфера ввода-вывода для устройства в байтах. N — десятичное число: для АЦПУ — от 15 до 255, для терминала — от 2 до 255.

Команда SET FILE устанавливает маркер конца файла, включая в каталог синоним, что дает возможность обращаться к файлу с использованием нескольких имен, удаляет из каталога ссылку на файл, освобождает незанятые блоки файла. Ф о р м а т :

SET FILE спецификация-файла/ключ

К л ю ч и :

/ENTER: синоним-файла — вводит дополнительную спецификацию, используя которую можно обращаться к файлу;

/NOWARNINGS — запрещает выдачу сообщений об ошибках в процессе выполнения команды;
/REMOVE — удаляет синоним из каталога, который был введен с помощью ключа /ENTER. Если по ошибке удалена единственная ссылка на файл в каталоге, следует использовать программу проверки файловой структуры VFY для восстановления;
/TRUNCATE — позволяет усекать файлы до их реальных размеров, освобождая неиспользованные, но зарезервированные блоки;
/END_OF_FILE[:(BLOCK: N, BYTE: M)] —определяет конец файла. Если не заданы параметры BLOCK и BYTE, то конец файла фиксируется за последним битом последнего блока файла, N определяет номер блока, где фиксируется конец файла. Если все байты последнего блока использованы, конец файла может быть зафиксирован в первом байте следующего блока. M — указывает номер (начиная от 0) первого свободного байта, где может быть размещен маркер конца файла.

Команда SET GROUPFLAGS создает или удаляет групповые глобальные флаги событий. Непривилегированный пользователь может оперировать только с флагами своей группы. Ф о р м а т :

SET GROUPFLAGS: N/ключ

где N — номер группы, для которой создаются или уничтожаются флаги.

К л ю ч и :

/CREATE — требует создания групповых глобальных флагов. /CREATE — значение по умолчанию;
/DELETE — удаляет групповые глобальные флаги.

Команда SET HOST позволяет логически подключить терминал пользователя к удаленному узлу. Локальный и удаленный узлы должны быть связаны между собой физической линией связи и на обоих узлах должна функционировать система программного обеспечения (СПО) «Сеть СММ» или совместимое с ней ПО. После подключения к удаленному узлу пользователь должен пройти на нем процедуру регистрации. Для прекращения сеанса работы с удаленным узлом используется команда LOGOUT. Ф о р м а т :

SET HOST имя-удаленного-узла

где имя-удаленного-узла — идентифицирует узел, к которому необходимо подключиться.

Команда SET LIBRARY/DIRECTORY переназначает системный библиотечный каталог.
Ф о р м а т :

SET LIBRARY/DIRECTORY: [каталог]

Команда SET [NO] PARTITION — привилегированная команда, она создает или удаляет раздел. Если в разделе установлена задача, общая область или загружен драйвер, его нельзя удалить. Ф о р м а т :

SET [NO] PARTITЮN:имя-раздела/ключ

где имя-раздела — имя раздела (от 1 до 6 символов), который должен быть создан или удален (SET NOPARTITION).

К л ю ч и :

/BASE:N — устанавливает базовый адрес раздела в 64-байтных блоках;
/DEVICE — определяет раздел для доступа к регистрам устройств;
/SIZE:M — задает размер раздела в 64-байтных блоках;
/SYSTEM — определяемый раздел является системно-управляемым;
/TOP:аргумент — позволяет перемещать верхнюю границу системно-управляемого раздела. Аргумент может принимать следующие значения:
TN — перемещает верхнюю границу раздела вверх на N 64-байтных блоков, увеличивая раздел;
— N — уменьшает раздел на N 64-байтных блоков;
N — устанавливает раздел размером N 64-байтных блоков;
[+]∗— перемещает верхнюю границу раздела вниз, насколько возможно.

Команда SET PASSWORD позволяет непривилегированному пользователю изменять пароль. Необходимо, чтобы в системе была установлена задача ... PSW. Привилегированный пользователь использует задачу ACNT. Ф о р м а т :

SET PASSWORD

Команда работает в диалоговом режиме и в результате следуют запросы:

OLD PASSWORD : старый-пароль
NEW PASSWORD : новый-пароль
VERIFICATION : повторный ввод нового пароля

Вводимые пользователем ответы не отображаются на экране.

Команда SET PRIORITY — привилегированная, она изменяет приоритет выполнения указанной задачи. Ф о р м а т :

SET PRIORITY: N имя-задачи

где имя-задачи — задает имя активной задачи, приоритет выполнения которой изменяется на N (1<N<250).

Команда SET PROTECTION изменяет код защиты существующего файла (файлов). Непривилегированный пользователь может изменять код защиты файлов, которые хранятся в его каталоге. Ф о р м а т ы :

SET PROTECTION (код) спецификация-файла/ключ
SET PROTECTION спецификация-файла/ключ (код)

где спецификация-файла — спецификация файла (файлов), защита которых изменяется. Спецификации нескольких файлов отдаляются друг от друга запятыми. Допускает применение символов маскирования;

код — должен всегда задаваться в круглых скобках. Может изменять защиту для всех 4 категорий пользователей (SYSTEM, OWNER, GROUP, WORLD), которые применяются в ОСРВМ. Для каждой категории могут быть разрешены следующие операции: чтение — R, запись — W, расширение — E, стирание — D.

П р и м е р задания кода:

(SYSTEM: RWED, OWNER: RWED, GROUP: **RWED, WORLD: R)**

Полный набор разрешенных операций — RWED, усеченный R. Он означает только чтение.

К л ю ч и :

/DATE: DD-MMM-YY— код защиты изменяется только для файлов с указанной датой создания;

/EXCLUDE: спецификация файла — при изменении защиты файлов исключить из операции файлы, заданные спецификацией в данном ключе;

/SINCE: DD-MMM-YY — код защиты изменяется для файлов с датой создания, начиная с указанной;

/THROUGH: DD-MMM-YY — код защиты изменяется для файлов с датой создания до указанной, включая ее;

/TODAY — код защиты изменяется для файлов, которые созданы сегодня.

Команда SET PROTECTION/[NO] DEFAULT устанавливает код защиты по умолчанию для всех файлов, созданных после ввода этой команды. Команда действует только во время текущей терминальной сессии. SET PROTECTION/NODEFAULT задает код защиты по умолчанию, который определен ранее как код защиты но умолчанию для тома. Ф о р м а т ы :

SET PROTECTION: (код) /DEFAULT
SET PROTECTION/NODEFAULT

где код — код защиты, устанавливаемый по умолчанию для вновь создаваемых файлов.

Формат — такой же, как в команде SET PROTECTION.

Команда SET QUEUE/ENTRY изменяет атрибуты задания или файлов в очереди в подсистеме диспетчера очередей. Атрибуты активного задания не изменяются. Ф о р м а т :

SET QUEUE/ENTRY: N/ключ(и)

К л ю ч и :

/AFTER: (DD-MMM-YY HH: MM) — устанавливает дату и время, до которого задача блокируется в очереди. Допускается не указывать дату, тогда берется текущая. Если не указывать время, берется 00:00 указанной даты;

/COPIES: N — устанавливает число копий (N) файла, содержащегося в задании для вывода на печать. Данный ключ должен использоваться совместно с ключом /FILE_POSITION: N;

/[NO] DELETE — разрешает или запрещает удаление файла после распечатки. Используется только с ключом /FILE_POSITION:N;

/FILE_POSITION:N — указывает номер файла (N) в задании для вывода на печать. Используется совместно с ключами /COPIES и /[NO] DELETE. Для определения N следует использовать команду SHOW QUEUE;

/FORMS: N — устанавливает новый номер формата для задания на печать;

/JOBCOUNT:N — устанавливает количество копий задания (N) при выводе на печать. Для указания количества копий файла в задании используется ключ /COPIES;

/LENGTH: N — определяет количество строк (N), которые будут распечатаны на одной странице;

/LOWERCASE — требует, чтобы задание распечатывалось на АЦПУ, для которого действует ключ /LOWERCASE;

/PAGE_COUNT: N — задает максимальное количество (N) печатных страниц задания;

/PRIORITY: N — устанавливает приоритет задания (N< 150 — для непривилегированного пользователя, N<250 — для привилегированного пользователя);

/RELEASE — освобождает задание, задержанное в очереди ожидания;

/[NO] RESTART — устанавливает атрибут перезапуска прерванного задания. /RESTART — прерванное задание запускается вновь, /NORESTART — с точки останова;

/UPPERCASE — направляет задание на АЦПУ, для которого действует ключ /UPPERCASE.

Команда SET QUEUE/JOB изменяет атрибуты всего задания, стоящего в очереди, или файлов, входящих в задание. Ф о р м а т :

SET QUEUE/JOB имя-очереди имя-задания

где имя-очереди — имя очереди заданий на печать; имя-задания — имя задания, атрибуты которого изменяются.

К л ю ч и :

/AFTER: (DD-MMM-YY HH:MM)
/COPIES: N
/[NO] DELETE
/FILE_POSITION: N
/FORMS: N
/JOBCOUNT: N
/LENGTH:N
/LOWERCASE
/PAGE_COUNT:N
/PRIORITY:N
/RELEASE
/[NO] RESTART
/UPPERCASE

Ключи команды SET QUEUE/JOB идентичны ключам команды SET QUEUE/ENTRY.

Команда SET SYSTEM — привилегированная команда, которая устанавливает некоторые характеристики системы. Ф о р м а т :

SET SYSTEM /ключ (и)

К л ю ч и :

/DIRECTORY: [каталог] — устанавливает заданный каталог как системный;

/EXTENSION_LIMIT:N — задает максимальный размер, до которого могут быть расширены задачи по директиве EXTK $ (N — длина в 64-байтных блоках);

/[NO] LOGINS — разрешает или запрещает регистрацию пользователей в системе;

/NETWORK_UIC: [G, M] — определяет каталог, в котором размещается сетевое программное обеспечение;

/PACKETS:N — задает максимальное число требований ввода-вывода, которые могут быть поставлены в очередь (0<N<15). По умолчанию N = 5;

/POOL:вершина:максимум:общий-размер — увеличивает размер системной динамической памяти.

где вершина — начало системной динамической памяти в 64-байтных блоках;
максимум — максимальный размер в словах (десятичное число);
общий-размер — общий размер системной динамической памяти в словах (десятичное число);

/POOL/LIMITS:аргумент — устанавливает предельные характеристики системной динамической памяти, используемые задачей PMT.

где HIGH = N — задает верхний предел системной динамической памяти в байтах. По умолчанию HIGH=1600;

LOW=N — устанавливает нижний предел системной динамической памяти в байтах. По умолчанию LOW=600;

MINIMUM SIZE = N — определяет минимальный размер в байтах самого большого свободного блока системной динамической памяти. По умолчанию MINIMUM_SIZE = 200;

/TASK_PRIORITY=N — устанавливает максимальный приоритет непривилегированной задачи, с которым она может конкурировать за память, когда системный пул находится в «низком состоянии». По умолчанию TASK_PRIORITY=51.

Команда SET TERMINAL позволяет устанавливать характеристики терминала. Непривилегированные пользователи могут устанавливать характеристики своего терминала, привилегированные — любого. Ф о р м а т :

SET TERMINAL [: TTNN:]/ключ(и)

К л ю ч и общего назначения:

/[NO]BRODCAST — разрешает (/BRODCAST) или запрещает вывод на терминал сообщений от других пользователей;

/CLI = имя-CLI — задает имя интерпретатора командных строк для терминала (от 1 до 6 символов в коде RADIX-50);

/[NO]CONTROL = C — определяет реакцию системы на ввод с клавиатуры CTRL/C. При использовании /CONTROL = C задача завершается аварийно. /NOCONTROL = C приводит к выводу подсказки DCL>;

/DCL — устанавливает DCL в качестве интерпретатора командных строк;

/[NO]HOLD_SCREEN — управляет режимом поэкранного вывода;

/INQUIRE — требует выполнения автоматической идентификации терминала и установки присущих ему характеристик;

/ [NO] LOWERCASE — определяет, будут ли символы, вводимые в нижнем регистре, преобразовываться в символы верхнего регистра;

если используется /LOWERCASE — не будут;

/MCR — устанавливает MCR в качестве интерпретатора командных строк;

/[NO]PRIVILEGED — привилегированный ключ. Позволяет управлять статусом привилегированности терминала;

/SPEED:(скорость-передачи, скорость-приема)— определяет скорость приема и передачи (в бодах);

/[NO] UPPERCASE — управляет преобразованием символов, введенных с терминала в нижнем регистре. /NOUPPERCASE — не выполняется преобразование символов, введенных в нижнем регистре;

/WIDTH: N — устанавливает длину строки терминала (16<N<255).

К л ю ч и установки терминала:

/[NO] ADVANCED_VIDEO — устанавливает или не устанавливает для терминалов типа VT100 следующие характеристики: мерцание экрана, поэкранный вывод, длина строки — 132 символа;

/[NO] ANSI_CRT — определяет, поддерживает ли терминал набор специальных символов ANSI./ANSI_CRT, кроме того, приводит к игнорированию некоторых ESC-последовательностей;

/[NO]AUTOBAUD — требует автоматического определения скорости линии связи; по умолчанию — /NOAUTOBAUD;

/[NO] BLOCK MODE — определяет, будет ли установлен для терминала блочный режим передачи данных;

/CRFILL: N — определяет число символов-заполнителей, которые выводятся после <CR> (0<N<7);

/[NO]DEC_CRT — указывает, будет ли терминал совместим с VT100 снизу вверх;

/[NO] EDIT_MODE — указывает, может ли терминал выполнять дополнительные функции редактирования;

/[NO] FORMFEED — определяет способ перевода формата, /FORMFEED— аппаратный перевод формата;

/[NO] HARDCOPY— определяет, позволяет ли терминал работать с твердой копией;

/[NO]HOSTSYNC — разрешает или запрещает блокировку клавиатуры при заполнении терминального буфера;

/[NO]LFFILL — указывает, будет ли терминальный драйвер добавлять четыре символа-заполнителя при обработке символа вертикальной табуляции, перевода строки или формата;

/PAGE_LENGTH: N — определяет количество строк на экране (1<N< <255);

/PRINTER_PORT — указывает на наличие печатающего устройства у терминала;

/[NO] REGIS — определяет возможность терминала выполнять графические команды;

/[NO] SCOPE — определяет, будет ли терминал установлен как дисплей;

/[NO] SOFT_CHARACTERS — разрешает или запрещает обработку символов, определенных пользователем;

/[NO] TAB — указывает на наличие или отсутствие аппаратной обработки символов горизонтальной табуляции;

/[NO] TRANSLATIONROUTINE [:аргумент] — позволяет связать терминал (или прекратить связь) с программой преобразования символов, указанной «аргументом»;

/[NO]TTSYNC — определяет реакцию системы на управляющие символы CTRL/S и CTRL/Q. По умолчанию управляющие символы обрабатываются (/TTSYNC).

К л ю ч и установки драйвера:

/[NO] ECHO — разрешает или запрещает эхо-отображение на терминале; по умолчанию — /ECHO;

/[NO] EIGHT_BIT — разрешает или запрещает работу с 8-битным кодом; по умолчанию — /NOEIGHT_BIT;

/[NO] ESCAPE — разрешает или запрещает прием ESC-последовательности с терминала; по умолчанию — /NOESCAPE;

/[NO] FULL_DUPLEX — определяет режим работы терминала как дуплексный (/FULL_DUPLEX) или полудуплексный;

/[NO] INTERACTIVE — указывает, будут ли вводимые символы обрабатываться терминальным драйвером до передачи пользовательской задаче; по умолчанию - /INTERACTIVE;

/[NO] LOCAL — указывает, подключен ли терминал к телефонной линии; по умолчанию — /LOCAL;

/[NO] PARITY: [{ ODD / EVEN }] — указывает наличие контроля по паритету для терминальной линии.

Если контроль выполняется, следует указывать тип контроля: ODD — по нечетности, EVEN — по четности;

/[NO]PASSALL — указывает, будут ли символы обрабатываться терминальным драйвером до передачи их пользовательской задаче; /PASSALL — не будут обрабатываться и при этом автоматически устанавливаться /EIGHT_BIT;

/[NO] REMOTE — определяет, подключен ли терминал к телефонной линии;

/[NO] PASTHRU — определяет, как драйвер терминала будет обрабатывать специальные символы; по умолчанию — /NOPASTHRU. При этом драйвер сам обрабатывает все специальные символы (<CR>, <LF>, CTRL/S и т. д.). Если задано /PASTHRU, специальные символы передаются задаче пользователя, кроме CTRL/S и CTRL/Q;

/[NO] SERIAL — определяет возможность параллельного выполнения задач. По умолчанию — /NOSERIAL — параллельное выполнение разрешено;

/[NO] SLAVE — управляет статусом подчиненности терминала, по умолчанию — /NOSLAVE — терминал неподчиненный;

/[NO]TYPEAHEAD [:N] — управляет возможностью терминального драйвера сохранять введенные символы в буфере до передачи их задаче пользователя. N позволяет установить размер терминального буфера. В системах с разделенными областями инструкций и данных 0<N<255 (по умолчанию N = 86, в других системах N = 58);

/[NO] WRAP — управляет возможностью драйвера генерировать < CR > и < LF >, когда число символов превышает число, заданное ключом /WIDTH; по умолчанию — /WRAP.

Команда SET UIC устанавливает UIC по умолчанию. Для привилегированных пользователей одновременно изменяется UIC по защите, если для терминала установлено NONAMED _ DIRECTORY. Если установлено NAME_DIRECTORY, то изменяется только UIC по защите, а UIC по умолчанию не изменится. Ф о р м а т :

SET UIC [G, M]

Команда SHOW ACCOUNTING выводит статистику о работе системы. Ф о р м а т :

SHOW ACCOUNTING

Команда подробно описана в разд. 6.

Команда SHOW ASSIGNMENTS выводит на терминал локальные и логические регистрационные назначения. Ф о р м а т :

SHOW ASSIGNMENTS

К л ю ч и :

/ALL — выводит на терминал все локальные, регистрационные и групповые назначения логических имен, а также все глобальные назначения. Чтобы
вывести назначения для терминала TTNN:, данный ключ используется совместно с ключом /TERMINAL: TTNN:;

/GLOBAL — указывает, что на терминал будут выведены все глобальные логические назначения, сделанные в системе;

/GROUP [: G] — указывает, что на терминал будут выведены групповые логические имена для указанного номера группы. Привилегированный пользователь может вывести назначения для любых групп. Ключ имеет смысл только для систем, поддерживающих расширенные логические имена;

/LOCAL — указывает, что на терминал будут выводиться локальные и регистрационные назначения, сделанные с данного терминала. Значение ключа /LOCAL принимается по умолчанию;

/LOGIN — имеет то же значение, что и ключ /LOCAL. Ключ привилегированный;

/SYSTEM — имеет то же значение, что и ключ /GLOBAL;

/TERMINAL: TTNN: — указывает, что на терминал будут выводиться локальные и регистрационные назначения для терминала TTNN:; ключ привилегированный.

Команда SHOW CACHE выводит системную информацию о кэшировании данных. Ф о р м а т :

SHOW CACHE [DDNN:]

Подробная информация о команде приведена в разд. 6.

Команда SHOW CLOCK_QUEUE выводит на терминал информацию о задачах, находящихся в очереди запросов запуска по времени. Ф о р м а т :

SHOW CLOCK_QUEUE

В ответ выводится информация, содержащая имена задач, времена запуска и интервал передиспетчеризации.

Команда SHOW COMMON выводит имена разделяемых общих областей, установленных в системе, адреса их блоков управления разделом, количество присоединенных задач и состояние каждой области. Ф о р м а т :

SHOW COMMON [:имя-области]

где имя-области — имя области, информацию о которой необходимо распечатать. По умолчанию выводится информация о всех областях.

К л ю ч :

/TASK — указывает, что следует вывести список задач, присоединенных к указанной области, а также число, которое означает количество отображений задачи в область.

Команда SHOW [DAY]TIME выводит па терминал установленные в системе дату и время. Ф о р м а т :

SHOW DAY TIME

Команда SHOW DEFAULT выводит на терминал текущие умолчания для устройства и каталога, информацию о работе с именованными или неименованными каталогами, а также UIC пользователя по защите. Ф о р м а т :

SHOW DEFAULT

Команда SHOW DEVICES выводит на терминал информацию об устройствах, включенных в систему. Ф о р м а т :

SHOW DEVICES

К л ю ч и :

/DD [NN:] — выводит информацию о всех устройствах указанного типа (DD — мнемоническое имя устройства);
/[NO] PUBLIC — выводит список всех устройств, установленных как общие или личные;
/SYSTEM — то же значение, что и /PUBLIC;
/WIDTH:DDNN: — выводит размер буфера ввода-вывода для указанных устройств.

Команда SHOW GROUPFLAGS выводит групповые глобальные флаги событий, используемые в системе. Ф о р м а т :

SHOW GROUPFLAGS

Команда SHOW HOST выводит на терминал имя узла сети, с которым логически связан терминал, а также имя и номер версии ОС, загруженной в этом узле. Ф о р м а т :

SHOW HOST

Команда SHOW LIBRARY выводит на терминал текущий библиотечный каталог. Ф о р м а т :

SHOW LIBRARY

Команда SHOW LOGICALS имеет те же функции и ключи, что и SHOW ASSIGNMENTS.

Команда SHOW MEMORY вызывает задачу RMD, которая представляет на терминале динамическую картину распределения памяти в системе. Ф о р м а т :

SHOW MEMORY

В ответ выводится следующая информация: тип и версия операционной системы, имя узла, размер памяти, текущая дата и время, имя выполняющейся задачи, количество свободных блоков на первых четырех дисках с файловой структурой, информация о динамическом нуле, информация о вторичном пуле, информация о разделах, число задач в памяти и их размер, имя каждой задачи, общей области или драйвера и их расположение в памяти, размер и расположение раздела, последовательный счетчик ошибок (ERL). При выводе используется следующая символика:

D — системно-управляемый раздел;
P — вторичный пул или динамическая память;
< > — активная задача;
[] — задача не активна, но занимает память;
!! — именованный раздел;
+ + — неименованный раздел;
( ) — загруженный драйвер;
— — — — задача не фиксирована в памяти;
= = = — задача фиксирована в памяти;
E — управляющая программа.

Строки звездочек пропорциональны количеству памяти, занимаемому каждым разделом (одна звездочка соответствует 1 Кслову).

Команда SHOW PARTITIONS выводит на терминал информацию о разделах системы. Ф о р м а т :

SHOW PARTITIONS [: имя]

где имя — имя раздела, о котором выводится информация; если оно не указано, выводится информация о всех разделах.
Состав выводимой информации:

имя-раздела адрес-PCB базовый-адрес размер тип

где базовый адрес и размер выводятся в 64-байтных блоках. Если раздел содержит подразделы, о каждом выводится информация в виде:

адрес-PCB базовый-адрес размер тип содержимое

где тип раздела может иметь следующие значения;

MAIN — главный раздел;
TASK — динамический подраздел, содержащий задачу;
DRIVER — раздел драйвера устройства;
RO COM — общий раздел с доступом только для чтения;
RW COM — общий раздел с доступом для чтения и записи;
DEVICE — раздел внешней страницы памяти;
SEC POOL — вторичный пул.

При выводе содержимого раздела используется символика команды SHOW MEMORY.

Команда SHOW PRIVILEGES выводит список привилегированных терминалов, зарегистрированных в системе. Ф о р м а т :

SHOW PRIVILEGES

Команда SHOW PROCESSOR выводит на терминал информацию о процессоре пакетной обработки, печатающих и других устройствах, находящихся под управлением диспетчера очередей (гл.6). Ф о р м а т :

SHOW PROCESSOR [имя-процессора]

Если параметр не указан, выводится информация о всех процессорах.

К л ю ч и :

/BATCH — указывает, что выводится информация о процессоре пакетной обработки;
/DEVICE, /PRINT — указывают, что выводится информация о процессорах вывода на печать.

Команда SHOW PROTECTION выводит на терминал пользовательский код защиты файла по умолчанию. Ф о р м а т :

SHOW PROTECTION

Команда SHOW QUEUE выводит на терминал информацию об очередях вывода на печать и пакетной обработки заданий. Ф о р м а т :

SHOW QUEUE [имя]

где имя — имя очереди, о которой требуется вывести информацию, например PRINT — главная очередь вывода на печать.

К л ю ч и :

/ALL — указывает, что будет выводиться информация о всех очередях;
/BATCH — указывает, что будет выводиться информация только об очередях пакетной обработки;
/BRIEF — указывает, что информация будет выводиться в кратком формате, включающем имя очереди, назначения для очереди и имена заданий, содержащихся в очереди;
/DEVICE, /PRINT — указывают, что будет выводиться информация только об очередях вывода на печать;
/ENTRY:N — указывает, что будет выводиться информация только о задании с номером N;
/FILES — указывает, что будет выводиться информация об очередях, назначениях, заданиях в очереди и файлах, составляющих задания. Этот ключ принимается по умолчанию;
/FORMS [:N] — указывает, что будет выводиться информация только о заданиях, установленных с указанной формой. Если аргумент не указан, будет выводиться информация обо всех заданиях с формой, отличной от ( );
/FULL — указывает, что будет выводиться полная информация, включая имя очереди, назначения, задания, характеристики заданий и файлы, составляющие задания;
/NAME: имя-задания — указывает, что будет выводиться информация только о задании с данным именем;
/OWNER_UIC: код-идентификации-пользователя — определяет, что будет выводиться информация только о заданиях указанного пользователя.

Команда SHOW SYSTEM выводит на терминал текущую информацию о системе. Ф о р м а т :

SHOW SYSTEM

К л ю ч и :

/CLI — выводит на терминал информацию о каждом интерпретаторе командных строк, содержащую имя CLI, имя задачи, обеспечивающей его работу, количество терминалов и флаги состояния CLI: ACT — активен, DCB — неработоспособен, PRV — доступен только привилегированным пользователям, RST —действие ограничено;
/DIRECTORY — выводит на терминал текущий системный каталог;
/EXTENSION_LIMIT — выводит на терминал максимальный размер задачи в 32-словных блоках, которого она может достичь;
/NETWORK_UIC — выводит текущий каталог для функционирующего сетевого программного обеспечения СЕТЬ СММ;
/PACKETS — выводит на терминал максимальное и текущее число доступных пакетов ввода-вывода;
/POOL — выводит текущее состояние системного пула, указывая: адрес первого доступного для пользователя раздела, размер наибольшего свободного блока и общего размера пула в десятичных словах;
/POOL/LIMITS — выводит текущие значения следующих предельных параметров пула: максимальный и минимальный размеры пула в байтах, минимальный размер в байтах наибольшего свободного блока пула, низший приоритет, который может иметь непривилегированная задача, чтобы конкурировать за память в «низком» пуле;
/SECONDARY_POOL — выводит на терминал информацию о вторичном пуле, указывая размер вторичного пула и свободного вторичного пула в 32-словных блоках, а также процентное содержание свободных блоков.

Команда SHOW TASKS/ACTIVE выводит на терминал информацию об активных задачах в системе. Ф о р м а т :

SHOW TASKS [:имя-задачи] /ACTIVE [TTNN:]

где TTNN: — имя терминала, для которого выводится информация об активных задачах в кратком формате.
Если имя задачи не указано, выводится информация обо всех активных задачах.

К л ю ч и :

/ALL — указывает, что будет выводиться информация обо всех активных задачах в системе, без этого ключа будет выводиться информация только о задачах, запущенных с данного терминала;

/BRIEF — указывает, что информация выводится в кратком формате, включающем имя задачи и терминала;

/FULL — ключ вывода в полном формате, по которому выводятся; имя задачи, адрес блока управления задачей, имя раздела, адрес блока управления разделом, границы раздела, приоритет, флаги состояния задачи, номер терминала TI, счетчики ввода-вывода и буферизованного ввода-вывода, флаги локальных событий, содержимое регистров и PSW.

Команда SHOW TASKS/DYNAMIC вызывает системную задачу RMD для вывода текущего состояния одной или всех активных задач.

Ф о р м а т 1 (для вывода информации об одной задаче):

SHOW TASKS; имя-задачи/DYNAMIC

К л ю ч :

/RATE:N — определяет скорость изменения информации на экране; N — число секунд между сменами кадра.

Ф о р м а т 2 (для вывода информации о нескольких активных задачах):

SHOW TASKS/ACTIVE/DYNAMIC

К л ю ч и :

/OWNER: аргумент — определяет устройство, с которого запущены задачи, информация о которых должна быть получена. Значения аргументов:

DDNN; — имя устройства, псевдоустройства, логическое имя или номер терминала;

ALL — будет выводиться информация обо всех задачах в системе;

/PRIORITY:N — указывает, что будет выводиться информация о задачах, имеющих данный приоритет и меньше;

/RATE: N — см. формат 1.

Если в процессе выполнения команды необходимо изменить устройство, приоритет или скорость изменения информации, следует нажать клавишу < ESC > и ввести нужное значение.

Команда SHOW TASKS/INSTALLED выводит на терминал информацию об установленных задачах. Ф о р м а т :

SHOW TASKS [: имя-задачи]/INSTALLED

К л ю ч и :

/BRIEF — указывает, что информация будет выводиться в кратком формате, который включает: имя задачи, версию, имя раздела, в котором задача установлена, приоритет, размер задачи в байтах, устройство, с которого задача была загружена, номер логического блока дискового адреса задачи, состояние задачи в памяти — фиксирована или временно выгружена. Этот ключ используется по умолчанию;

/DEVICE: DDNN: — определяет, что будут выводиться имена и состояния всех задач, установленных с указанного устройства;

/FULL — указывает, что информация об установленных задачах выводится в полном формате аналогично выводу в команде SHOW TASKS/ ACTIVE/FULL.

Команда SHOW TASKS/LOGICAL_UNITS выводит на терминал список физических устройств, статически назначенных задаче, и соответствующие им логические номера устройств. Ф о р м а т :

SHOW TASKS:имя-задачи/LOGICAL_UNITS

Список назначений LUN не включает динамические назначения. Задача не должна быть установлена командой RUN.

Команда SHOW TERMINAL выводит информацию о терминалах системы. Ф о р м а т :

SHOW TERMINAL [:TTNN:]

где TTNN: — номер терминала, о котором запрашивается информация. Если номер не указан, информация выводится о текущем терминале; если опущены ключи, выводятся все атрибуты терминала.

К л ю ч и :

Существует группа специфических ключей для определенных типов терминалов:

/HT, /RT — выводят список сетевых терминалов;
/TI — аналогично использованию команды без ключей;
/TT — выводит список функционирующих терминалов;
/VT — выводит список виртуальных терминалов.

Все основные ключи команды SHOW TERMINAL связаны с характеристиками терминалов, установленными по команде SET TERMINAL, где они и описаны. Ниже представлено соответствие между ключами команд SHOW и SET TERMINAL и обозначениями, используемыми в распечатке.

Ключ команды SHOW TERMINAL	Распечатка команды SHOW TERMINAL	Ключ команды SET TERMINAL
ADVANCED_VIDEO	AVO	ADVANCED_VIDEO
ANSI__CRT	ANSI	ANSI_CRT
AUTOBAUD	ABAUD=	AUTOBAUD
BLOCK_MODE	BLKMOD	BLOCK_MODE
BROADCAST	BRO	BROADCAST
CLI: CLINAME	CLI=	CLI: CLINAME
CONTROL=C	CTRLC	CONTROL = C
CRFILL	HFILL=	CRFILL
DEC_CRT	DEC	DEC_CRT
DCL	CLI = DCL	DCL
ECHO	ECHO	ECHO
EDIT_MODE	EDIT	EDIT_MODE
EIGHT_BIT	EBC	EDIT_MODE
ESCAPE	ESC	ESCAPE
FORM_FEED	FORM	FORM_FEED
FULL_DUPLEX	FDX	FULL_DUPLEX
HARDCOPY	NOCRT	HARDCOPY NOSCOPE
HOSTSYNC	HSYNC	HOSTSYNC
LEFILL	VFILL	LEFILL
LOCAL	NOREMOTE	LOCAL NOREMOTE
LOWERCASE	LOWER	LOWERCASE
MCR	CLI = MCR	MCR
MODEL	TERM=	MODEL
PAGE_LENGTH	LINES =	PAGE_LENGTH
PASSALL	RPA	PASSALL INTERACTIVE
PASTHRU	PASTHRU	PASTHRU
PRIVILEGE	PRIV=	PRIVILEGE
PRINTER_PORT	PRINTER_PORT	PRINTER_PORT
REGIS	REGIS	REGIS
REMOTE	REMOTE	REMOTE NOLOCAL
SCOPE	CRT	SCOPE NOHARDCOPY
SERIAL	SERIAL	SERIAL
SLAVE	SLAVE=	SLAVE
SOFT_CHARACTERS	SOFT	SOFT
SPEED	SPEED=	SPEED
TAB	HHT	TAB
TTSYNC	TTSYNC	TTSYNC
TYPEAHEAD	TYPEAHEAD	TYPEAHEAD
UPPERCASE	NOLOWER	UPPERCASE
WIDTH	BUF=	WIDTH
WRAP	WRAP	WRAP

Команда SHOW UIC выводит на терминал код идентификации пользователя (UIC). UIC идентифицирует пользователя для системы и определяет его привилегированность. Ф о р м а т :

SHOW UIC

Команда SHOW USERS выводит на терминал список зарегистрированных в системе пользователей с номером терминала,
с которого пользователь зарегистрирован, каталог и UIC по умолчанию. Ф о р м а т :

SHOW USERS

Команда START возобновляет выполнение задачи, остановленной по директиве STOP$S. Ф о р м а т :

START [имя-задачи]

где имя-задачи — имя активизируемой задачи. Если оно не указано, начинает выполняться задача, загруженная по команде RUN.

К л ю ч :

/TERMINAL: TTNN: — разрешает привилегированному пользователю возобновлять выполнение задачи, запущенной с указанного терминала.

Команда START/PROCESSOR запускает процессор вывода на печать или процессор пакетной обработки подсистемы диспетчера очередей (подробно команда описывается в разд. 6). Ф о р м а т :

START /тип-процессора имя-процессора .

Команда START/QUEUE запускает указанную очередь (подробно команда описывается в разд. 6). Ф о р м а т :

START/QUEUE имя-очереди

где имя-очереди — имя очереди задания вывода на печать или очереди обработки пакетных заданий.

Команда START/QUEUE/MANAGER запускает диспетчер очередей. Ф о р м а т :

START/QUEUE/MANAGER

Команда START/UNBLOCK возобновляет выполнение задачи, блокированной по команде STOP/BLOCK. Ф о р м а т :

START/UNBLOCK [имя-задачи]

Если параметр не указан, активизируется выполнение задачи, заблокированной с терминала пользователя.

К л ю ч :

/TERMINAL:TTNN: — указывает, что будет разблокирована задача, выполняющаяся с указанного терминала.

Команда STOP/ABORT немедленно завершает выполнение текущего задания. Остановленное задание удаляется из очереди и начинает обрабатываться следующее задание, привилегированные пользователи могут остановить любое задание, непривилегированные—только их собственное. Ф о р м а т :

STOP/ABORT устройство-печати

где устройство-печати — спецификация устройства, на которое выводится текущее задание.

Команда STOP/BLOCK блокирует выполнение установленной задачи, понижая ее приоритет до нуля. Ф о р м а т :

STOP/BLOCK [имя-задачи]

Если параметр не указан, блокируется задача, выполняемая с терминала пользователя.

К л ю ч :

/TERMINAL:TTNN: — указывает, что будет блокирована задача, выполняющаяся с указанного терминала.

Команда STOP/PROCESSOR приостанавливает работу процессора вывода на печать или процессора пакетной обработки (подробно команда описана в разд. 6). Ф о р м а т :

STOP/QUEUE имя-очереди

Команда STOP/QUEUE/MANAGER останавливает работу диспетчера очередей после обработки текущего задания. Ф о р м а т :

STOP/QUEUE/MANAGER

К л ю ч :

/ABORT — прекращает работу диспетчера очередей немедленно.

Команда SUBMIT ставит в очередь на выполнение пакетные задания. Ф о р м а т :

SUBMIT спецификация файла

Ключи описаны в разд. 6.

Команда TYPE выводит указанные файлы на терминал пользователя. Ф о р м а т :

TYPE спецификация-файла

где спецификация-файла — спецификация одного или нескольких файлов, выводимых на терминал.

К файлам должен быть доступ по чтению. Пользователи могут употреблять стандартные символы маскирования «*» и «%».

К л ю ч и :

/DATE:DD-MMM-YY— определяет, что выводятся файлы только с указанной датой создания;

/EXCLUDE: спецификация-файла — определяет, что из вывода исключаются указанные файлы;

SHARED — означает, что возможен доступ к файлу во время его вывода;

/SINCE:DD-MMM-YY, /THROUGH:DD-MMM- YY — определяют, что будут выводиться файлы с датой создания, начиная с указанной (/SINCE) и до указанной включительно;

/TODAY — указывает, что будут выводиться файлы с текущей датой создания;

/NOWARNING — отменяет вывод сообщений об ошибках.

Команда UNFIX освобождает память, занятую фиксированной задачей или районом. Ф о р м а т :

UNFIX имя-задачи

К л ю ч и :

/REGION — указывает, что освобождается общий район, а не задача;

/READONLY_SEGMENT указывает, что освобождается район типа «только чтение» многопользовательской задачи.

Команда UNLOCK разблокирует заблокированные файлы, т. е. файлы, которые были некорректно закрыты. Ф о р м а т :

UNLOCK спецификация-файла

К л ю ч и :

/DATE, /EXCLUDE, /SINCE, /THROUCH, /TODAY, /NOWARNING — соответствуют ключам команды TYPE.

4.4. КОМАНДНЫЕ ПРОЦЕДУРЫ

Командные процедуры предназначены для автоматизации деятельности пользователя системы ОСРВМ.

Командные процедуры реализуются с помощью командных файлов. В ОСРВМ различают командные файлы задач и интерпретатора командных строк CLI. Первые содержат информацию в терминах соответствующей задачи, вторые — в терминах процессора командных файлов.

В настоящем разделе рассматриваются командные файлы, интерпретируемые процессором командных файлов (далее процессором). Эти файлы содержат последовательность команд CLI и, возможно, директив процессора.

Командный файл — это текстовый файл типа CMD (по умолчанию) или другого типа. Для активизации командного файла необходимо ввести командную строку:

@ имя-командного-файла. CMD [/ключ]

или

@ имя-командного-файла [/ключ]

или

@ имя-командного-файла. тип [/ключ] (если тип командного файла отличен от CMD).

Процессор командных файлов (АТ.) считывает указанный файл и интерпретирует каждую строку или как команду, адресованную CLI, или как собственную директиву. Каждой директиве предшествует точка.

Процессор допускает использование следующих ключей в командной строке:

/[NO]TR — разрешает или запрещает (/NOTR) отладочную распечатку последовательно выполняемых директив процессора (по умолчанию /NOTR);

/[NO]DE — требует или запрещает удаление командного файла после выполнения. Если выполнение файла прервано по STOP или «/», то он в любом случае не стирается;

/[NO]MC — передает или не передает команды, не опознанные процессором, интерпретатору командных строк (по умолчанию /MC);

/[NO]CLI — синоним ключа /[NO]MC (по умолчанию /CLI);

/LB: модуль — указывает, что файл является библиотекой командных процедур типа CLB, а модуль — той процедурой, которая должна быть выполнена.

В ОСРВМ имеется стандартная библиотека командных процедур LB: [1, 2] INDSYS.CLB, которая содержит следующие процедуры:

INDCFG — выводит параметры построения выполняющегося процессора;

INDDMP — выводит на терминал аварийную распечатку содержимого таблицы переменных процессора;
1NDPRF — выполняет разбор спецификации файла;
INDSFN — формирует информацию о конфигурации системы;
INDVFY — выводит значение всех специальных переменных;
QIOERR — формирует строковое расширение кодов ошибок <FILERR>.

Пример:

@ [1, 2]INDSYS/LB:QIOERR

/[NO]LO — разрешает или запрещает использование вызываемым командным файлом локальных переменных вызывающего командного файла (по умолчанию /NOLO).

Набор директив процессора представляет собой процедурный язык командных файлов. Процессор оперирует с тремя типами переменных: логическими, числовыми и строковыми. Логические принимают значение «истина» или «ложь», числовые — целочисленные значения от 0 до 65535 (10), строковые включают от 0 до 132 (10) байт в символьном коде.

Числовые переменные и константы могут объединяться с другими переменными или константами в выражения путем использования знаков арифметических операций: +, —, ∗, /.

Логические выражения образуются с помощью операций «И» (.AND.) «ИЛИ» (.OR.) «НЕ» (.NOT.), исключающее «ИЛИ» (!) и могут присваиваться логической переменной.

Строковые выражения образуются из строковых констант, переменных или подстрок посредством строкового оператора « + » и могут присваиваться строковым переменным.

Тип переменной определяется первой директивой, которая присваивает ей значение. Последующие директивы могут изменить значение переменной, но не ее тип.

Процессор автоматически определяет специальные переменные в зависимости от системных характеристик и других факторов.

Имена всех переменных должны начинаться с букв (от A до Z) или знака $; остальные символы — буквы, цифры или знак $. Длина имени переменной — не более 6 символов.

Имена специальных переменных обязательно заключаются в угловые скобки (например, <EOF>).

Процессор использует также резервные переменные для передачи параметров командного файла. Используются следующие локальные переменные: P1, P2, ..., P9, COMMON. Последняя включает всю информацию, содержащуюся в командной строке, в том числе спецификацию командного файла. Переменные P1 — P9 содержат элементы командной строка, ограниченные пробелами.

4.4.1. ДИРЕКТИВЫ ПРОЦЕССОРА КОМАНДНЫХ ФАЙЛОВ

Директивы отделяются от аргументов хотя бы одним пробелом или знаком горизонтальной табуляции. На каждой строке может быть записана только одна директива. Метки, применяемые для изменения порядка следования директив, располагаются в начале строки и заканчиваются двоеточием. Допустима только одна метка в строке.

Ниже приводится описание директив процессора командных файлов.

Директива .ASK выводит на терминал указанную в команде строку символов (вопрос) и в зависимости от ответа оператора присваивает логической переменной ssssss значение «истина» или «ложь». Если разрешено использование тайм-аута и если он задан в директиве, процессор АТ. ждет ответа не более промежутка времени, заданного тайм-аутом, после чего присваивает переменной значение по умолчанию.

Форматы:

.ASK ssssss строка-символов
.ASK [значение-по-умолчанию: тайм-аут] ssssss строка-символов
.ASK [:тайм-аут] ssssss строка символов

Таймаут задается в следующем виде:

NM

где N — величина тайм-аута;
M — единица измерения этой величины — H (часы), M (минуты), S (секунды), T (тики).

Форматы ответов пользователя на вопрос директивы .ASK: Y<CR> присваивает ssssss значение «истина», N<CR> — «ложь», <CR> — присваивает ssssss значение по умолчанию или, если оно не

задано, «ложь», <ESC> — «истина».

П р и м е р :

.ASK III DO YOU WANT TO INSTALL PIP

Процессор АТ. добавляет к вопросу символ «□» в начале строки и знак «?» — в ее конце вместе с вариантами ответа, т. е.

>∗DO.YOU WANT TO INSTALL PIP [Y/N]:

Директива .ASKN позволяет присвоить переменной ssssss числовое значение. При использовании второй формы АТ. проверит также, лежит ли введенное значение в интервале «верхняя граница — нижняя граница».

Ф о р м а т ы :

.ASKN ssssss строка-символов
.ASKN [нижняя-граница:верхняя-граница:
значение по умолчанию: тайм-аут] ssssss строка-символов

П р и м е р :

.ASKN [3: 36: 26: 30S] AA DEFINE NUM AA

Процессор АТ. выведет строку:

>∗ DEFINE NUM AA [O R: 3—36 D: 26 T: 30S]:

где 0 — определяет тип переменной АА (восьмеричный по умолчанию);

R: 3—36 — допустимый интервал значений АА;
D: 26 — значение по умолчанию;
T; 30S — тайм-аут.

Директива .ASKS позволяет ввести с терминала строку символов и присвоить строковой переменной ssssss соответствующее значение. Ф о р м а т ы :

.ASKS ssssss строка-символов
.ASKS [MIN-длина: MAX-длина: строка-по-умолчанию: тайм-аут]
ssssss строка-символов

П р и м е р :

.ASKS T ENTER TIME >∗ ENTER TIME [S];

где S — тип переменной — строковая.

Директива .BEGIN обозначает начало блока вида .BEGIN — .END. Метки и локальные переменные, определенные внутри блока, являются неопределенными вне блока.

Директива .CHAIN продолжает обработку, употребляя новый командный файл. Ключи (/ключ), которые могут быть заданы в директиве .CHAIN, аналогичны ключам, описанным при активизации командного файла. Ф о р м а т :

.CHAIN имя-косвенного-командного-файла/ключ

Директива .CLOSE закрывает вторичный файл, открытый директивой .OPEN. Ф о р м а т :

.CLOSE [#N]

#N — необязательный параметр, он указывает номер закрываемого файла. 0<N<k—1, где k — количество открытых файлов.

Директива .DATA позволяет записать одну строку символов длиной до 132(10) в предварительно открытый по директиве .OPEN вторичный-файл. Ф о р м а т :

.DATA [#N] строка-символов

Если открыто несколько файлов, то # N определяет, в какой (по порядку открытия) из них следует записать строку символов.

П р и м е р :

.SETS ABC «BEGIN» , .OPEN KK .DATA ’ABC’ .CLOSE

В данном примере в файл KK.DAT записывается одна строка, содержащая слово BEGIN.

Директива .DEC вычитает 1 из числовой переменной ssssss. АТ. аварийно завершится, если ssssss

описана как строковая или логическая переменная. Ф о р м а т :

.DEC ssssss

Директива .DELAY приостанавливает выполнение командного файла на указанное параметром N число часов, минут, секунд, тиков. Ф о р м а т :

$$.\text{DELAY N} \left\{ \begin{matrix} \text{H} \\ \text{M} \\ \text{S} \\ \text{T} \end{matrix} \right\}$$

Директивы .ENABLE и .DISABLE разрешают или запрещают использование заданных в директиве параметров. Ф о р м а т ы :

.ENABLE параметр [,параметр] ...
.ENABLE DATA [#N]
.DISABLE параметр [,параметр] ...

П а р а м е т р ы :

ATTACH — присоединение терминала при отображении комментариев. По умолчанию —.ENABLE ATTACH;

CONTROL/Z — использование CTRL/Z в качестве ответа на вопросы по директивам типа .ASKX. Если этот параметр запрещен, CTRL/Z рассматривается как конец файла и AT. заканчивает работу. По умолчанию — .DISABLE CONTROL/Z;

DATA — вывод данных во вторичный файл. Строки, следующие за .ENABLE DATA, записываются в ранее открытый вторичный файл (#N определяет, в какой именно) до появления директивы .DISABLE DATA;

GLOBAL — использование глобальных символов. .ENABLE GLOBAL позволяет рассматривать все символы, начинающиеся со знака $ как глобальные, определенные на всех уровнях вложенности командных файлов. По умолчанию —.ENABLE GLOBAL;

DECIMAL — интерпретация всех цифровых символов по умолчанию как десятичных, .DISABLE DECIMAL — как восьмеричных. По умолчанию — .DISABLE DECIMAL;

DELETE — стирание командного файла после выполнения. По умолчанию — .DISABLE DELETE — стирание запрещено;

DISPLAY — вывод полей директивы .ASKX и @ <EOF>;

MCR — передача всех командных строк, не опознанных AT., DCL, CLI пользователя MCR;

CLI — синоним команды MCR;

LOWERCASE — преобразование русских букв в латинские. По умолчанию—.ENABLE LOWERCASE;

SUBSTITUTION — замена. Если замена разрешена. AT. заменяет строковые переменные, заключенные в апострофы, на их значения. По умолчанию .DISABLE SUBSTITUTION запрещает замену;

ESCAPE — использование <ESC> в качестве ответа на .ASK. <ESC> присваивает соответствующей переменной значение «истина». По умолчанию .DISABLE ESCAPE запрещает .AT принимать <ESC> как ответ на .ASK;

ESCAPE-SEQ — разрешение .AT принимать ESC — последовательность с терминала. .DISABLE ESCAPE-SEQ — запрещение приема. По умолчанию — .DISABLE ESCAPE-SEQ;

OVERFLOW — использование отрицательных значений числовых переменных. .DISABLE OVERFLOW — запрещение отрицательных значений — действует по умолчанию;

QUIET — отказ от эхо-сопровождения команд CLI, выданных из командного файла. По умолчанию — .DISABLE QUIET;

TIMEOUT — использование тайм-аутов в директивах типа .ASKX, .ENABLE TIMEOUT — разрешено использование тайм-аутов. Действует по умолчанию;

TRACE — трассировка. .DISABLE TRACE — значение по умолчанию;

TRUNCATE — усечение строк. Если усечение разрешено, процессор AT. игнорирует ошибки, связанные с превышением допустимой длины строки — 132 символа. .DISABLE TRUNCATE — значение по умолчанию.

П р и м е р :

.ENABLE SUBSTITUTION .ASKS NAM ENTER FILE NAME MAC ’NAM’=’NAM’

При выполнении командного файла:

>∗ENTER FILE NAME [S]: AABB
>MAC AABB = AABB

Директива .ERASE уничтожает все локальные и глобальные определения символов или указанное определение глобального символа. Ф о р м а т ы :

.ERASE LOCAL
.ERASE GLOBAL
.ERASE SYMBOL глобальный-символ

Если директива .ERASE LOCAL задана внутри блока .BEGIN — .END, то она уничтожает только локальные символы данного блока. Автоматически уничтожаются определения локальных символов при использовании файла другого уровня вложенности.

Директива EXIT завершает обработку в командном файле данного уровня и передает управление в файл более высокого уровня. Ф о р м а т :

EXIT [значение]

Значение необязательного параметра присваивается специальной переменной <EXSTAT>.

Директива .GOSUB вызывает запоминание состояния обработки командного файла (для последующего возврата) и выполняет переход на метку, заданную в директиве .GOSUB. Эта метка может располагаться даже внутри блока .BEGIN — .END. Метка в директиве .GOSUB указывается без ведущей точки и не сопровождается двоеточием (например, .GOSUB MET1).

Подпрограмма, на которую передается управление по директиве .GOSUB, должна заканчиваться директивой .RETURN. Ф о р м а т :

.GOSUB метка [параметры]

где параметры, если они имеются, становятся значением специальной переменной— <COMMAN>. Для разбора параметров следует использовать директиву .PARSE.

Директива .GOTO — безусловный переход на метку. Процессор AT. ищет метку для выполнения перехода только внутри того блока .BEGIN —.END, в котором стоит .GOTO. Таким образом, нельзя передать управление в другой блок или выйти за пределы данного. Ф о р м а т :

.GOTO метка

П р и м е р :

GOTO 100

Директива .IF инициирует проверку значения логического выражения, заданного в директиве. Если значение выражения — «истина», то выполняется директива 1, в противном случае анализируется следующая строка. Ф о р м а т :

.IF логическое-выражение директива-1

Логическое выражение может включать стандартные операции сравнения: EQ, NE, GE, LE, GT, LT.

При сравнении строковых переменных анализируются попарно символы слева направо. При первом несовпадении операция заканчивается.

Директива .IFACT/.IFNACT позволяет проверить, активна задача или бездействует. При соблюдении условия выполняется директива, находящаяся в той же строке. Ф о р м а т ы :

.IFACT имя-задачи директива-1
.IFNACT имя-задачи директива-1

П р и м е р :

.IFACT PIP.GOTO 300

Директива .IFDF/.IFNDF проверяет, определена ли логическая, строковая или числовая переменная. Ф о р м а т ы :

.IFDF ssssss директива-1
.IFNDF ssssss директива-1

Директива .IFINS/.IFNINS осуществляет проверку, установлена ли указанная в директиве задача. Если проверка успешна, выполняется директива, заданная в той же строке. Ф о р м а т ы :

.IFINS имя-задачи директива-1
.IFNINS имя-задачи директива-1

П р и м е р :

.IFNINS PIP INS $PIP

Директива .IFENABLE/.IFDISABLE. Ф о р м а т ы

.IFFNABLE параметр директива-1
.IFDISABLE параметр директива-1

Параметры рассмотрены при описании директивы ENABLE. Кроме того, используются следующие аргументы:

.FULL_DUPLEX — полнодуплексный терминальный драйвер включен в систему;
LOCAL — ключ /LO указан в командной строке;

POTASK — поддержка порождающих и порожденных задач включена в систему.

Директива .IFLOA/.IFNLOA проверяет, находится ли в оперативной памяти указанный загружаемый драйвер. В результате проверки может быть выполнена директива 1. Ф о р м а т ы :

.IFLOA DD: директива-1
.IFNLOA DD: директива-1

П р и м е р :

IFNLOA LP: LOA LP:

Директива .IFT/.IFF проверяет значение указанной логической переменной. Ф о р м а т ы :

.IFT ssssss директива-1
.IFF ssssss директива-1

где директиа-1, указанная в строке, выполняется, если ssssss имеет значение «истина» (1-я строка) или «ложь» (2-я строка).

П р и м е ч а н и е . Несколько директив типа .IF могут быть использованы для проведения сложной проверки путем объединения их с помощью логических функций .OR (ИЛИ) и .AND (И). Другие обозначения ! и &.

П р и м е р :

.IFT A.OR(.IFT B.AND .IFF C) .GOTO C

Директива .INC увеличивает числовую переменную ssssss на единицу. Ф о р м а т :

.INC ssssss

Директива .ONERR позволяет передать управление на подпрограмму обработки ошибок, выявленных при работе процессора AT. Директива применяется только для обработки следующих ошибок:

задача не установлена (.XQT, .WAIT);
переменная не определена;
синтаксические ошибки (.XQT, .WAIT, .DELAY);
ошибочная команда;
ошибка замены;
ошибка в типе переменной (.IF, .IFT, .IFF, .INC, DEC);
некорректная попытка изменения типа переменной (ASK, .SETX);
ошибка при работе с файлом.

Ф о р м а т :

.ONERR метка-1

Подпрограмма, начинающаяся меткой-1, работает со специальной переменной <ERRCTL>. Такая подпрограмма должна завершаться выдачей сообщения и директивой .EXIT.

Директива .OPEN открывает вторичный файл для записи. Ф о р м а т :

.OPEN [#N] спецификация-файла

где N — номер-открываемого-файла.
Тип файла по умолчанию — .DAT, номер — О.

Директива .OPENA открывает вторичный файл для дополнения. Данные добавляются в конец существующего файла. Ф о р м а т :

.OPENA [#N] спецификация-файла

где N — номер-открываемого-файла.

Директива .OPENR открывает вторичный файл для чтения. Ф о р м а т :

.OPENR [#N] спецификация-файла

где N — номер-открываемого-файла.

Директива .PARSE позволяет произвести разбивку командной строки на требуемое количество подстрок. Ф о р м а т :

.PARSE строковая-переменная строка-управления переменная-1 ... переменная N

где строковой переменной должно быть присвоено значение командной строки, которую следует разбить на подстроки;

строка управления — это набор знаков-разделителей, заключенных в кавычки. Разделители (чаще всего — запятые или пробелы) отделяют друг от друга параметры в командной строке или команду от параметров. Если разделителей указано меньше, чем имеется параметров в строке, последний разделитель используется многократно;

переменная 1 ... переменная N — это строковые переменные, которые получают значения подстрок в результате разбивки. Переменная получает значение «пусто», если соответствующая подстрока пуста. Специальная переменная <STRLEN> после разбивки содержит действительное число подстрок в строковой переменной.

П р и м е р :

.PARSE <COMMAN> «,» A B C D E

Пусть <COMMAN> содержит AAA 1,37,,DFC тогда в результате разбивки переменная A содержит AAA, B—1, C—37, D — «пусто», E — DFC.

Директива .PAUSE прекращает выполнение косвенного командного файла до момента вмешательства оператора. После вмешательства интерпретация возобновляется с точки приостановки. Ф о р м а т :

.PAUSE

Директива .READ считывает очередную запись из открытого файла, заданного параметром #N (см. выше директиву .OPENX этого раздела), и присваивает строковой переменной ssssss содержимое этой записи. Специальный символ <EOF> принимает значение «истина», если обнаружен конец файла. <FILERR> содержит код ошибки файловой системы ОСРВМ Ошибка может быть зафиксирована в процессе операции «чтение». Ф о р м а т :

.READ [#N] ssssss

Директива .RETURN реализует возврат из подпрограммы. Ф о р м а т :

.RETURN

Директива .SETT/.SETF/.SETL определяет или изменяет значение логической переменной ssssss. Если до директивы .SETX переменная не была определена, то директива определит ее и присвоит значение «истина» (.SETT), «ложь» (.SETF) или то значение, которое определит логическое выражение LLLLLL. Директива AT. завершится аварийно, если ssssss была ранее определена как числовая или строковая переменная. Ф о р м а т ы :

.SETT ssssss
.SETF ssssss
.SETL ssssss LLLLLL

П р и м е р :

SETL A B .OR C

Директива .SETN определяет или изменяет значение числовой переменной. Ф о р м а т :

.SETN ssssss числовое-выражение

П р и м е р :

.SETN ABC 3∗(B7—5)

Директива .SETO/.SETD переопределяет или определяет впервые числовую переменную как восьмеричную (.SETO) или десятичную (.SETD) с необходимым пересчетом значения. Ф о р м а т :

.SETO ssssss
.SETD ssssss

П р и м е р :

.SETN A 10 Присваивает A значение 10 (B)
.S E T D A Объявляет A десятичной переменной со знаком 8(10)
.SETO A Объявляет A восьмеричной переменной со знаком 10 (8)
Процессор AT. завершится аварийно, если ssssss ранее была определена как строковая или логическая переменная.

Директива .SETS переопределяет значение строковой переменной или задает ее вновь и присваивает ей указанное значение. Ф о р м а т :

. SETS ssssss строковое-выражение

где строковое выражение может включать конкатенацию строковых переменных и строковых констант. Переменные могут входить в выражение полностью или частично. Строковые константы заключаются в кавычки ("). Знак конкатенации — плюс (+). Если необходимо включить в выражение часть строковой переменной, применяется обозначение вида

ssssss [M: N]

где M — номер символа строковой переменной, начиная с которого следует извлекать символы для формирования константы;

N — номер последнего символа. Вместо N можно указать □, что означает включить до конца.

П р и м е р :

.SETS Al "ABCDEF"; присвоить A1 значение ABCDEF
.SETS A2 Al [3:4] A2 имеет значение "CD"
.SETS A3 Al [1:3] + "KM" + A2; A3→ABCKMCD

Директивы .STOP и /немедленно прекращают обработку командного файла AT. Процессор выводит сообщение EOF. Ф о р м а т :

.STOP [числовое-выражение]

Числовое выражение передается процессору AT. в качестве кода завершения. Директива / вызывает те же действия, что и .STOP без параметра.

Директива .TEST. Применяются два формата этой директивы. Ф о р м а т 1 :

.TEST ssssss

Директива .TEST анализирует ssssss и в результате присваивает специальным символам следующие значения:

1) если ssssss — строковая переменная, то <SYMTYP> присваивается значение 4, <STRLEN> содержит длину строки. Кроме того, переменные <ALPHAN>, <NUMBER>, <RAD50>, <OSTAL> получают значения в зависимости от характеристик ssssss;
2) если ssssss — числовая переменная, <SYMTYP> присваивается значение 2;
3) если ssssss—восьмеричная переменная, <SYMTYP> присваивается значение 2, а <OCTAL> —«истина»;
4) если ssssss — логическая переменная, <SYMTYP> получает значение 0.

Ф о р м а т 2 :

.TEST строка подстрока

где строка — строковая переменная или константа; подстрока — строковое выражение.

Процессор AT. ищет подстроку внутри строки. Если находит, то <STRLEN> указывает на номер знака строки, с которого начинается подстрока. В противном случае STRLEN «обнуляется».

Директива .TESTDEVICE позволяет получить информацию о любом устройстве системы. Соответствующая информация становится значением строковой переменной <EXSTRI>. Элементы этой строки разделены запятыми, что облегчает анализ с помощью директив .PARSE и .TEST. Ф о р м а т :

.TESTDEVICE DD[NN];

где DD[NN]: — физическое имя устройства.

В результате выполнения данной директивы переменная <EXSTRI> имеет следующий формат:

DDNN: XX1, XX2, XX3, XX4, атрибут 1 . . . атрибут N

где XX1 . . . XX4 — содержимое полей U.CW1 . . . U.CW4 из UCB устройства.

А т р и б у т ы :

NSD — такого устройства нет в системе;
LOD — драйвер загружен;
UNL — драйвер не загружен;
ONL — устройство в рабочем состоянии;
OFL — устройство в автономном режиме;
MTD — устройство смонтировано;
NMT—устройство не смонтировано или это немонтируемое устройство;
FOR — том смонтирован как созданный на другой системе;
NFO — устройство немонтируемое или том не смонтирован как созданный на другой системе;
PUB — общее устройство;
NPO — не общее устройство;
ATT — устройство присоединено к другой задаче;
ATU — устройство присоединено к данному AT.;
NAT—устройство не присоединено;
ALO — устройство зарезервировано для другого терминала;
ALU — устройство зарезервировано данным терминалом;
NAL — устройство свободно.

Директива .TESTFILE имеет два формата.

Ф о р м а т 1 :

.TESTFILE LLNN:

где LLNN: — логическое имя устройства. Процессор AT. находит связанное с ним физическое устройство и помещает его имя в <FILSPC>, а код ошибки системы управления файлами — в <FILERR>.

Ф о р м а т 2 :

.TESTFILE спецификация-файла

Данная директива формирует в переменной <FILSPC> полную спецификацию файла, указанного в директиве .TESTFILE.

П р и м е р :

.TESTFILE AA: INN. TXT
<FILERR> = 1 — успешное завершение
<FILSCP> = DK1: [100, 100] INN. TXT; 17

где AA: — логическое имя устройства, назначенное на устройство DK1:.

Директива .TESTPARTITION позволяет получить информацию об указанном разделе. Ф о р м а т :

.TESTPARTITION имя-раздела

Строковая переменная <EXSTRI> содержит в результате выполнения данной директивы следующую информацию: имя раздела, базу, длину, тип. База и длина измеряются в 64-байтных блоках. Если искомого раздела нет в системе, то вместо типа передается NSP.

Директива .TRANSLATE дает возможность использовать в командном файле расширенное значение логических имен. Оно присваивается им строковой переменной <EXSTRI>. Ф о р м а т :

.TRANSLATE ([N]) логическое-имя

где N — число от 1 до 10, определяющее количество итерационных трансляций логического имени. Квадратные скобки — элемент синтаксиса; ∗— трансляция выполняется требуемое число раз; логическое имя — имя, которое должно быть расширено.

Специальная переменная <EOF> получает значение «истина», если расширенное логическое имя является результатом окончательной трансляции логического имени.

П р и м е р :

```
DEL A=B     TRANSLATE [∗] D
DEL B=C     TRANSLATE D
DEL C=D
```

Первая директива присваивает переменной <EXSTRI> значение A, <EOF> получает значение «истина», вторая директива .TRANSLATE присваивает <EXSTRI> значение C, а <EOF> приобретает значение «ложь».

Директива .WAIT приостанавливает работу директивы AT. до завершения указанной задачи. Ф о р м а т :

.WAIT имя-задачи

Если имя задачи не указано, используется TTNN:, <EXSTAT> содержит код завершения задачи.

Директива .XQT позволяет запустить несколько задач, не дожидаясь завершения ранее запущенных. Обычно процессор AT. запускает задачу и ждет ее завершения. Ф о р м а т :

.XQT имя-задачи командная-строка-задачи

П р и м е р :

```
.XQT MAC A = A
.XQT PIP A.TSK; ∗/DE
```

4.4.2. РАБОТА С ПРОЦЕССОРОМ КОМАНДНЫХ ФАЙЛОВ В ИНТЕРАКТИВНОМ РЕЖИМЕ

Для вызова процессора следует ввести командную строку вида

@TI:

В результате последует подсказка

AT.>

В ответ на эту подсказку можно вводить любые директивы. Для распечатки значений специальных переменных следует разрешить режим замены (.ENABLE SUBSTITUTION) и ввести имя переменной в

ответ на подсказку процессора:

AT. >' < имя-специалыюй-переменной >'

П р и м е р :

Редактирование, удаление, вывод и форматирование файлов:

```
.ENABLE QUIET
.ENABLE SUBSTITUTION
.ASKS FILNAM введите имя файла
  .ASKS FILTYP введите тип файла
  EDIT 'FILNAME', 'FILTYP'
  .ASK A удалить старые версии
  .IFT A PURGE/KEEP:2 'FILNAM', 'FILTYP'
  SET FILE/TRUNCATE 'FILNAM'. 'FILTYP'
  .ASK DOC хотите запустить DOC
  .IFT DOC .GOSUB PROG
  .ASK B хотите листинг
  .IFF B .GOTO 100
   .GOSUB LIST
  PRINT/FORMS: 'C'/COPIES: 'D' 'FILNAM', 'FILTYP'
.100:
        .EXIT
.PROC: RNO 'FILTNAM' = 'FILNAM'
        .SET FILTYP "MEM"
        .ASK F хотите удалить старые версии с расширением MEM
        .IFT F PURGE/KEEP: 2 'FILNAM' .MEM
        SET FILE/TRUNCATE 'FILNAM', MEM; *
        .RETURN
.LIST:
        .ASKN C введите N формы .ASKN D количество копий .RETURN
```

П р и м е ч а н и е . Используемые в СМ ЭВМ терминалы СМ7209.05 или СМ7232 соответствуют типу VT 100.

5. Создание задач

5.1. ФУНКЦИИ ПОСТРОИТЕЛЯ ЗАДАЧ

Единицей работы в ОСРВМ является задача. Задача состоит из одного или нескольких объектных модулей, скомпонованных в загрузочный модуль, — *образ задачи.*

Процесс создания задачи осуществляется построителем задач (ТКВ), который обеспечивает три основные функции:

компоновку (связь) объектных модулей;

назначение адресов для образа задачи;

создание структур данных для задачи.

При компоновке объектных модулей ТКВ создает единый файл образа задачи, разрешает ссылки к глобальным символам, не определенным в модулях, и к библиотеке объектных модулей. Задача строится с виртуального нулевого адреса и благодаря аппаратуре диспетчера памяти может перемещаться в памяти.

Структуры данных, создаваемые при построении задачи, используются при ее установке и выполнении.

5.2. СТРУКТУРА ЗАДАЧИ

Файл образа задачи, создаваемый ТКВ, содержит (в простейшей форме) метку, занимающую группу блоков, заголовок задачи, образ памяти задачи. Для выгружаемых задач файл образа задачи может содержать область выгрузки.

Метка содержит данные, используемые при установке задачи в системе: имя задачи и параметры раздела, в котором она должна выполняться, размер задачи и ее приоритет, таблицу назначений логических устройств. При установке задачи эта информация используется для создания блока управления задачей и инициализации ее заголовка.

Заголовок задачи содержит информацию, которая используется управляющей программой при выполнении и выгрузке задачи. Задача загружается в память и выгружается на диск вместе с заголовком задачи. Метка не загружается. ТКВ создает и частично инициализирует заголовок задачи, а при установке задачи инициализируется остальная часть заголовка.

Память задачи включает скомпонованные объектные модули программ, т. е. коды инструкций и данные, а также стек задачи. Стек является областью памяти задачи, которая может использоваться для временного хранения информации и связи подпрограмм. Обращение к стеку осуществляется через указатель стека

Трансляция исходной программы и связь объектных модулей задачи определяются контекстом программных секций.

Программная секция — единица памяти задачи, содержащая коды инструкций и данные, обладает следующими характеристиками:

именем;

набором атрибутов, определяющих тип секций, режим доступа, размещение и возможность перемещения в памяти;

длиной, определяющей объем памяти, резервируемой для программной секции.

Программные секции создаются в явном виде с помощью директивы макроассемблера .PSECT или с помощью специальных операторов на языках высокого уровня (например, COMMON в программах на Фортране). Если программная секция не задается явно в программе, то при трансляции создается пустая программная секция.

Атрибуты программных секций и их значения приведены в табл. 5.1.

Таблица 5.1

Атрибут	Обозначение	Значение
Код доступа к секции	RW	Чтение и запись. Данные можно считывать из программной секции и записывать в нее
	RO	Только чтение. Данные можно считывать из программной секции, но записывать в нее нельзя

Продолжение

Атрибут	Обозначение	Значение
Код типа секции	D	Данные. Программная секция содержит данные
	I	Инструкция. Программная секция содержит инструкции и данные или только инструкции
Код области действия	GBL	Глобальный. Имя программной секции распознается за границами сегмента. Построитель задач распределяет объем памяти для такой программной секции исходя из ссылок за пределами определяющего сегмента перекрытия
	LCL	Локальный. Имя программной секции распознается только внутри определяющего сегмента перекрытия. Построитель задач распределяет объем памяти для программной секции исходя из ссылок только внутри определяющего сегмента
Код сохранения	SAV	Сохранение. Программная секция с атрибутом SAV размещается внутри корневого сегмента
Код размещения секции	CON	Конкатенация. Программные секции с одинаковыми именами объединяются. Их общий объем есть сумма длин всех программных секций с одинаковым именем
	OVR	Перекрытие. Программные секции с одинаковыми именами перекрывают друг друга. Размер общей памяти равен длине самой большой программной секции с этим именем
Код перемещаемости	REL	Перемещаемый. Базовый адрес программной секции вычисляется относительно виртуального адреса задачи
	ABS	Абсолютный. Базовый адрес программной секции не является перемещаемым и всегда равен нулю

5.2.1. ПРИВИЛЕГИРОВАННЫЕ ЗАДАЧИ

В ОСРВМ предусмотрены два класса задач: привилегированные и непривилегированные.

Привилегированные задачи обладают специальными правами доступа к памяти и внешним устройствам, которыми не обладают непривилегированные задачи. Привилегированная задача имеет, за некоторым исключением, доступ к подпрограммам и структурам управляющей программы, чего не имеет непривилегированная задача. У привилегированных задач есть доступ к внешней странице памяти, у непривилегированных задач его нет. Наконец, привилегированная задача может обходить системные средства защиты, тогда как непривилегированная задача лишена этой возможности.

Вследствие своих особых прав доступа привилегированные задачи могут оказать нежелательные действия на систему. Привилегированная задача с ошибками может воздействовать на управляющую программу или на системные структуры данных. Более того, ошибки, порожденные такой привилегированной задачей, могут быть скрытыми и трудно обнаруживаемыми. По этим причинам необходимо соблюдать осторожность при разработке и выполнении привилегированных задач.

Когда пользователь прекращает работу в системе, необходимо проверить, закончила ли привилегированная задача свое выполнение, так как команда оператора BYE не прекращает работу привилегированных задач в отличие от непривилегированных. В момент нахождения привилегированной задачи в режиме ядра не могут быть выполнены ни команда оператора BYE, ни другие задачи до тех пор, пока привилегированная задача не завершит свою работу в этом режиме. Однако после выхода привилегированной задачи из режима ядра команда оператора BYE выполнится, зарегистрировав выход пользователя из системы. Сама же привилегированная задача все еще будет выполняться до своего полного завершения.

Если в момент нахождения привилегированной задачи в режиме пользователя произойдет прерывание по ошибке, управляющая программа прекратит выполнение этой привилегированной задачи. Однако, если такое прерывание произойдет в момент нахождения привилегированной задачи в режиме ядра, произойдет аварийное завершение работы системы. Аварийное завершение системы может произойти и тогда, когда привилегированная задача, находящаяся в режиме пользователя и имеющая ссылки к управляющей программе, некорректно использует системные структуры данных.

Для построения привилегированной задачи необходимо использовать ключ /PR: N, где аргумент N определяет номер первой пары регистров APR, используемой ТКВ для распределения адресного пространства для привилегированной задачи. Этот аргумент является необязательным, его значение по

умолчанию равно 5. Аргумент может принимать значения 0, 4 и 5.

Когда указывается ключ /PR : 0, задача строится как привилегированная, но не отображается на управляющую программу или внешнюю страницу памяти. Задача отображается с нулевого виртуального адреса и может иметь размер 32 Кслова. Задача не имеет доступа к памяти управляющей программы и к внешней странице.

Когда указываются ключи /PR: 4 или /PR: 5, ТКВ резервирует регистр APR7 для отображения внешней страницы памяти. Кроме того, память управляющей программы становится доступной для задачи путем резервирования регистров APR, необходимых для отображения управляющей программы в виртуальное адресное пространство задачи.

Выбор между значениями аргумента 4 и 5 определяется размером управляющей программы. Если размер управляющей программы не превышает 16 Кслов, то можно определить значения аргумента равным 4. Задача, построенная с ключом /PR: 4, может иметь максимальный размер 12 Кслов и отображается с виртуального адреса 100000 (8).

Если управляющая программа имеет размер 20 Кслов, необходимо указать значение аргумента равным 5. Задача, построенная с ключом /PR: 5, может иметь размер 8 Кслов и отображаться с виртуального адреса 120000 (8).

Если отображение внешней страницы не требуется, то размер привилегированной задачи может быть увеличен на 4 Кслова. Чтобы сообщить ТКВ о том, что задача не нуждается в отображении на внешнюю страницу памяти, необходимо использовать ключ / —IP.

Задача, построенная с ключами /PR: 4 или /PR: 5, может иметь доступ к управляющей программе, системным структурам данных и внешней странице памяти. Она может использовать подпрограммы управляющей программы и выполнять операции ввода-вывода логических блоков данных на монтированных томах. Такая задача может также выдавать макрокоманду $SWSTK, чтобы перейти из режима пользователя в режим ядра, что позволяет задаче обращаться к управляющей программе или к системным структурам данных без опасения прерываний и изменения этих данных во время их использования.

Задаче, построенной с ключом /PR: 0, предоставлена возможность обхода защиты файлов, использования директивы изменения приоритета (ALTP$), выполнения любой директивы управляющей программы, определения имени устройства в директивах порождения задач, выполнения операций ввода-вывода на монтированном или на распределенном томе.

Привилегированная задача в I/D-системах, т. е. системах, где управляющая программа отображена в I/D-пространствах, может иметь обычное отображение или отображение в I/D-пространствах. Так как в I/D-системах управляющая программа занимает более 16 Кслов памяти, использование ключа /PR : 4 в таких системах не допускается.

5.2.2. МНОГОПОЛЬЗОВАТЕЛЬСКИЕ ЗАДАЧИ

Многопользовательская задача — это задача, которая разделяет часть кодов, предназначенных только для чтения («чистые» коды) с двумя или более копиями частей кодов, предназначенных как для чтения, так и для записи («нечистые» коды). Когда система впервые получает запрос на запуск такой задачи, копии обеих ее частей загружаются в физическую память. На протяжении всего времени выполнения задачи все последующие запросы на ее запуск приводят к дублированию в физической памяти только ее части «нечистых» кодов. Таким образом, многопользовательские задачи более эффективно используют физическую память.

Для построения многопользовательской задачи необходимо указать ключ /MU в спецификации файла образа задачи. ТКВ использует атрибуты доступа программных секций для определения того, в какую часть образа многопользовательской задачи их помещать. Для части задачи (она включает заголовок задачи и область стека), содержащей программные секции «нечистых» кодов, ТКВ выделяет регистры APR младших адресов, а для части «чистых» кодов — регистры APR старших адресов. В отличие от обычных задач в многопользовательской задаче часть, содержащая «чистые» коды, аппаратно защищена.

В многопользовательской I/D-задаче, в дополнение к тому, что она должна быть разделена на две части, содержащие программные секции с атрибутами доступа «только чтение» и «чтение и запись», каждая такая часть также должна быть разделена на две зоны — зону инструкций и зону данных.

Если ни одна из частей обычной многопользовательской задачи не содержит резидентных в памяти

перекрытий, ТКВ создает и размещает в заголовке задачи два блока окон. Во время установки задачи в системе процедура INSTALL инициализирует блоки окон следующим образом:

блок окна 0 описывает диапазон виртуальных адресов для части задачи, содержащей программные секции для чтения и записи. Эта область всегда содержит заголовок задачи;

блок окна 1 описывает диапазон виртуальных адресов для программной секции, предназначенной только для чтения. Если многопользовательская задача является задачей с перекрытиями, то ее часть, предназначенная только для чтения, должна состоять только из резидентных в памяти перекрытий. В многопользовательских I/D-задачах регистры APR D-пространства отображают программные секции, предназначенные для чтения и записи и только для чтения и имеющие атрибут «данные». Аналогично APR I-пространства отображают программные секции, предназначенные для чтения и записи и только для чтения и имеющие атрибут «инструкции». ТКВ создает четыре блока окон для отображения многопользовательской I/D-задачи. Блоки окон 0 и 1 отображаются в район 0, который содержит инструкции и данные для чтения и записи. Блоки окон 2 и 3 отображаются в район 1, который содержит инструкции и данные только для чтения.

5.2.3. ЗАДАЧИ С ИСПОЛЬЗОВАНИЕМ I/D-ПРОСТРАНСТВА

В ОСРВМ ТКВ может разбивать задачи пользователя на пространство инструкций и пространство данных (I/D-пространства), отображая задачи с использованием окон в I/D-системе. В ОСРВМ существует разница между обычными задачами и I/D-задачами. Обычная задача — это задача, которая не разбивается при отображении на пространство инструкции и пространство данных.

I/D-система может работать только на вычислительных комплексах с процессором, обеспечивающим эту возможность. Обычная задача может работать в I/D-системе, но I/D-задача не может работать в системе, которая не имеет этих возможностей.

Пространство данных задачи пользователя — это пространство, содержащее данные, к которым задача пользователя имеет доступ через APR D-пространства. Возможность разбиения задачи на I/D-пространства позволяет использовать 16 регистров APR для отображения задачи: 8 регистров APR для пространства данных и 8 регистров APR — для пространства инструкций. Максимально используя I/D-пространства, задача может иметь 64 Кслова виртуального адресного пространства. Если скомпоновать такую задачу с 32-Ксловной библиотекой режима супервизора, она сможет иметь 96 Кслов виртуального адресного пространства.

Для разделения данных и инструкций в задаче пользователя используется атрибут I/D в директиве языка макроассемблер .PSECT. Кроме того, в задаче можно использовать директивы CRAWS и CRRG$ для динамического создания и размещения района пространства данных.

Два поля идентифицируют I/D-задачу. В заголовке задачи байт со смещением H.DMAP идентифицирует задачу как задачу с отображением в D-пространство. В блоке управления задачи (TCB) в четвертом слове состояния задачи бит T4. DSP указывает системе, что задача имеет I/D-пространства.

Ниже представлено несколько вариантов изображения комбинаций I/D-задачи с I/D-пространствами и без них, а также распределение регистров APR в этих случаях.

I/D-задача	I/D-система	Распределение регистров APR
Нет	Да	APR I-пространства и APR D-пространства содержат одинаковые адреса перемещения
Нет	Нет	APR I-пространства содержат перемещения
Да	Да	APR I-пространства отображают область инструкций. APR D-пространства отображают область данных
Да	Нет	Невозможно

Обычная задача отображает свои виртуальные адреса в логические адреса как посредством набора регистров APR I-пространства, так и посредством набора регистров APR D-пространства. Это означает, что ТКВ не разделяет пространство инструкций и пространство данных и не дает системе возможность различать пространства, поэтому задача должна отображать свое логическое адресное пространство через оба набора APR, которые перекрывают друг друга.

Для I/D-задачи ТКВ разделяет данные и инструкции. Алгоритм обработки такой задачи ТКВ предопределяет физическое размещение заголовка задачи в начале задачи, при этом задача должна начинаться в I-пространстве. Затем ТКВ копирует заголовок в D-пространство, так как управляющая программа использует для управления задачей заголовок из D-пространства. Стек задачи также

располагается в D-пространстве. Если задача имеет внешний заголовок (ключ /XH), управляющая программа не только копирует заголовок D-пространства, но и помещает его в непрерывную область памяти непосредственно перед I-пространством задачи.

Окно 0 в I/D-задаче отображает корневой сегмент в I-пространство, окно 1 отображает часть корневого сегмента в D-пространство. ТКВ резервирует использование этих двух окон, допуская определять до 14 дополнительных окон для I/D-задачи. Разделяемые библиотеки с I/D-пространствами недопустимы.

5.3. ПОСТРОЕНИЕ ЗАДАЧ С ПЕРЕКРЫТИЯМИ

Структура задач с перекрытиями обеспечивает уменьшение размеров физической памяти или виртуального адресного пространства, требуемого задачами при их выполнении. Для создания задачи с перекрытиями необходимо разделить задачи на сегменты: единственный корневой сегмент, который находится постоянно в памяти, и необходимое число сегментов перекрытий. Каждый сегмент является единицей загрузки и состоит из набора программных секций. Сегменты, которые перекрывают друг друга, должны быть логически независимыми, т. е. не содержать взаимных ссылок. Возможны два типа сегментов перекрытий (в дальнейшем просто перекрытий): перекрытия, резидентные на диске, и перекрытия, резидентные в памяти.

Разделяемым ресурсом для *перекрытий, резидентных на диске,* является физическая память и виртуальное адресное пространство. Использование этого типа перекрытий уменьшает потребности задачи в физической памяти, но увеличивает накладные расходы для загрузки перекрытий в память.

Перекрытия, которые в процессе выполнения задачи постоянно находятся в физической памяти, называются *перекрытиями, резидентными в памяти.* Эти перекрытия загружаются с диска только при первом обращении к ним, а затем остаются в памяти на все время выполнения задачи. Для перекрытий, резидентных в памяти, разделяемым ресурсом является только виртуальное адресное пространство. Задачи с такой структурой перекрытий выполняются быстрее, чем задачи с перекрытиями, резидентными на диске, так как в этом случае не требуется загрузка перекрытий с диска (кроме первого обращения).

5.3.1. СТРУКТУРА ПЕРЕКРЫТИЙ, РЕЗИДЕНТНЫХ НА ДИСКЕ

Рассмотрим структуру задачи TSK с перекрытиями, резидентными на диске. Задача TSK строится из четырех вводных файлов. Каждый вводной файл состоит из единственного объектного модуля с тем же именем, что и файл.

Задача строится по команде ТКВ:

```
>TKB TSK = CNTRL, A, B, C
```

Модуль CNTRL для этой задачи определяется корневым сегментом, а модули A, B, C — логически независимыми сегментами перекрытий.

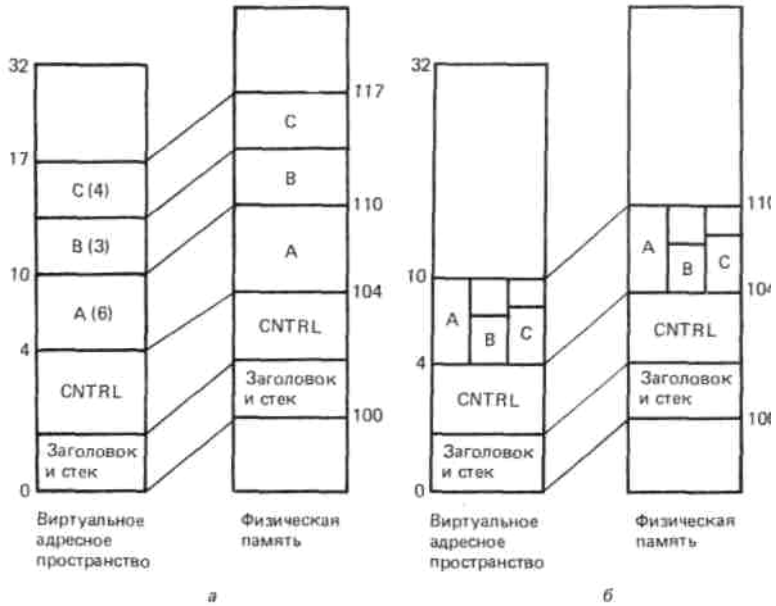

Рис. 5. Распределение памяти для односегментной (*а*) и многосегментной (*б*) задачи

Перекрытия A, B, C для такой задачи будут разделять между собой физическую память и виртуальное адресное пространство. Введение структуры перекрытий, резидентных на диске, уменьшает для задачи TSK. общую потребность в памяти.

Как видно из рис. 5, размер оперативной памяти, требуемой для задачи TSK как односегментной, определяется суммой длин всех модулей задачи — CNTRL, A, B, C.

Для задачи TSK, построенной как многосегментная задача с перекрытиями, резидентными на диске, требуемый размер оперативной памяти определяется длиной корневого сегмента CNTRL и длиной наибольшего сегмента перекрытий A. Кроме памяти, требуемой для размещения модулей, для многосегментной задачи необходима память для организации управления структурой перекрытий.

Разделяя сегменты перекрытий A, B, C на логически независимые части, можно еще больше сократить требуемый объем памяти для задачи TSK.

5.3.2. СТРУКТУРА ПЕРЕКРЫТИЙ, РЕЗИДЕНТНЫХ В ПАМЯТИ

Перекрытия, резидентные в памяти, загружаются с диска только при первом обращении к ним, а затем постоянно находятся в памяти, разделяя между собой виртуальное адресное пространство. В отличие от перекрытий, резидентных на диске, перекрытия, резидентные в памяти, не разделяют физическую память, так как они располагаются в отдельных областях, выравненных на границу 32-словного блока.

Процесс загрузки перекрытий, резидентных в памяти, связан (в отличие от собственно загрузки перекрытий, резидентных на диске) с отображением разделяемых виртуальных адресов в непрерывные области физической памяти, содержащие перекрытия.

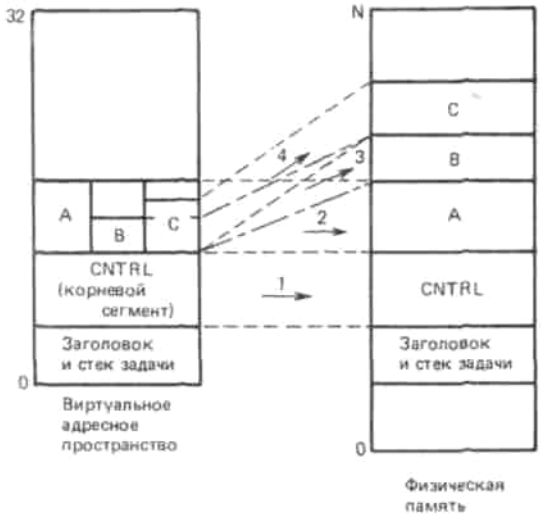

Рис. 6. Отображение сегментов перекрытий, резидентных в памяти, в различные (1, 2, 3 и 4) моменты времени

На рис. 6 представлено отображение сегментов перекрытий, резидентных в памяти.

Длина каждого сегмента перекрытий, резидентных в памяти, должна выбираться по возможности кратной 4 Кословам. Это определяется тем, что распределение виртуального адресного пространства, связанного с аппаратурой диспетчера памяти, осуществляется страницами по 4 Кслова.

В целях уменьшения объема требуемой физической памяти задачи общие библиотечные объектные модули целесообразно размещать в сегменте, доступном для всех ссылающихся сегментов.

Резидентные библиотеки в целях экономии виртуальной памяти целесообразно разбивать на сегменты, резидентные в памяти.

5.3.3. ДЕРЕВО ПЕРЕКРЫТИЙ

Структура задачи с перекрытиями может быть представлена в виде дерева перекрытий. Каждая ветвь дерева представляет собой сегмент. Параллельные ветви одного уровня соответствуют сегментам, которые перекрывают друг друга и должны быть логически независимыми. Примыкающие ветви соответствуют сегментам, которые не разделяют виртуальное адресное пространство или физическую память. Для этих сегментов не требуется логической независимости.

Построитель задач включает язык описания перекрытий для представления структуры задач с перекрытиями в виде одного или нескольких деревьев. Структура, содержащая несколько деревьев,

является «сложным» деревом. «Сложные» деревья перекрытий состоят из главного дерева и нескольких поддеревьев.

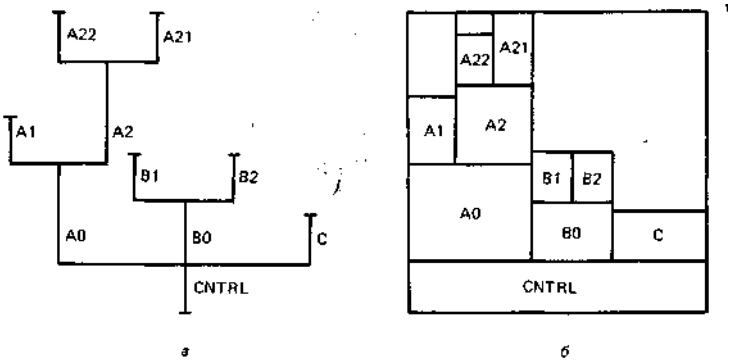

Рис. 7. Структура задачи с перекрытиями:

а — дерево перекрытий для задачи; *б* — диаграммы распределения памяти

Особенностью структуры сложного дерева является возможность доступа из одного поддерева ко всем модулям в главном дереве и других поддеревьях. Это объясняется тем, что каждое дерево размещается в собственной памяти.

На рис. 7 изображена структура задачи с перекрытиями, описываемая в виде единственного дерева перекрытий. Дерево имеет корневой сегмент CNTRL, три главные ветви A0, B0, C и шесть листьев — Al, A21, A22, B1, B2, C.

Дерево имеет столько ветвей, сколько у него листьев. Путь вниз определяется как путь от листа к корню (например, A22— A2—A0—CNTRL) путь вверх — как путь от корня к листу (например, CNTRL — A0 — A1). Понятия дерева и пути являются важными для понимания механизма загрузки и разрешения глобальных символов.

Модули могут вызывать другие модули, которые находятся на том же самом пути. На рис. 7 модуль CNTRL является общим для всех путей дерева и поэтому может вызывать любой модуль в дереве и быть вызванным из любого модуля. Если модуль одного сегмента перекрытий вызывает модуль из другого сегмента, этот сегмент должен находиться в памяти и быть отображенным или должен загружаться в память.

Процедура для разрешения глобальных символов в многосегментной задаче может быть описана следующим образом:

•построитель задач выбирает сегмент перекрытий для обработки в соответствии с размещением сегментов относительно корневого сегмента: обработка начинается с сегментов, наиболее удаленных от корня;

•каждый модуль в сегменте просматривается для поиска глобальных определений и ссылок;

•для каждого глобального определения символа построитель задач отыскивает ссылки на этот символ во всех сегментах на путях, проходящих через обрабатываемый сегмент; аналогичным способом для глобальных ссылок осуществляется поиск соответствующих определений;

•если глобальный символ определен несколько раз на разных путях и к нему имеется ссылка из сегмента, расположенного на общей части этих путей, символ считается неопределенным;

•если глобальный символ определен несколько раз на одном пути, символ считается многократно определенным;

•два или более определений глобального символа, находящихся на разных путях и не имеющих ссылок к ним из сегмента, общего для разных путей, являются допустимыми.

5.3.4. ЯЗЫК ОПИСАНИЯ ПЕРЕКРЫТИЙ

Язык описания перекрытий (ODL) построителя задач позволяет описывать структуру перекрытий задачи. Описание перекрытий является текстовым файлом, содержащим директивы языка описания перекрытий. Этот файл включает по умолчанию расширение ODL и указывается как единственный вводной файл в командной строке TKB с ключом /MP или в командной строке LINK с ключом /OVERLAY.

В языке ODL предусмотрено пять директив для описания дерева перекрытий: .ROOT, .END, .FCTR,

.NAME, .PSECT. Описание дерева может содержать только одну директиву .ROOT и одну директиву .END. Директива .END должна быть последней в описании дерева.

Директива .ROOT определяет структуру задачи с перекрытиями и обычно является первой в описании перекрытий. Аргументами директивы .ROOT могут быть спецификации файлов, имена сегментов, имена программных секций, имена библиотек и библиотечных модулей. Аргументы директивы .ROOT могут разделяться следующими операторами:

оператором «—» (тире), который указывает конкатенацию, т. е. что перекрытия должны находиться в физической памяти одновременно;

оператором «,» (запятая), который внутри скобок означает, что перекрытия разделяют физическую память или виртуальное адресное пространство;

оператор «!» (восклицательный знак), который перед левой скобкой указывает перекрытия, заключенные в скобки, как резидентные в памяти;

круглыми скобками, использующимися для ограничения группы сегментов, которые имеют одинаковый виртуальный адрес.

Директива .END должна быть последней. Она указывает построителю задач на завершение ввода описания перекрытия.

Описание перекрытий для задачи, представленное на рис. 7, может быть дано в следующем виде:

```
.ROOT CNTRL — (A0— (A1, A2 — (A22, A21)), B0— (Bl, B2), C)
.END
```

Директива .FCTR используется для представления описания перекрытий в более наглядном виде, что осуществляется разбиением строки директивы .ROOT на несколько строк. В директиве .FCTR употребляются такие же аргументы, как и в директиве .ROOT. Например, описание перекрытий, приведенное выше, с использованием директивы .FCTR может быть представлено так:

```
        .ROOT CNTRL— (AFCTR, BFCTR, C)
AFCTR:  .FCTR A0— (Al, A2 — (A21, A22))
BFCTR:  .FCTR B0— (Bl, B2)
.END
```

Директива .FCTR может быть вложенной; максимальная глубина вложения — 16.

Директива .NAME является средством для объявления имени сегмента и присвоения сегменту необходимых атрибутов. Она обеспечивает применимость метода автозагрузки для сегментов, не содержащих выполняемых инструкций, или для задания в явном виде имени сегмента, загружаемого методом ручной загрузки. Если директива .NAME не указывается, для идентификации сегмента используется имя первого файла или имя первой программной секции в сегменте. Формат директивы:

```
.NAME имя сегмента  [список атрибутов)
```

Здесь атрибуты разделяются запятой и являются необязательными. Значения атрибутов по умолчанию — NOGBL, DSK.

Атрибуты могут иметь следующие значения:

GBL —имя сегмента вводится в таблицу глобальных символов сегмента, что обеспечивает загрузку сегментов, не содержащих инструкций, методом автозагрузки;

NOGBL — имя сегмента не вводится в таблицу глобальных символов сегмента;

DSK — для заданного сегмента резервируется память на диске;

NODSK — для заданного сегмента не резервируется память на диске, что обеспечивает возможность отказа в выделении памяти в файле образа задачи для сегмента, не содержащего данных. Предполагается, что такой сегмент будет заполняться данными только при выполнении задачи.

Директива .PSECT является средством для управления размещением глобальной программной секции в требуемый сегмент. Имя программной секции должно быть указано первым в параметрах директивы .PSECT, а остальные параметры являются атрибутами программной секции, перечисленными в табл. 5.1. Они могут задаваться в произвольном порядке.

Для примера, приведенного в описании директивы .FCTR при размещении в корневом сегменте программной секции DATA с данными, описание перекрытий будет иметь вид:

```
        .PSECT DATA, RW, D, GBL, REL, OVR
        .ROOT CNTRL — DATA— (AFCTR, BFCTR, C)
AFCTR:  .FCTR A0 — (Al, A2 — (A21, A22))
BFCTR:  .FCTR B0 — (Bl, B2)
        .END
```

Такое размещение программной секции DATA обеспечивает информационное взаимодействие всех сегментов перекрытий через общую область данных в корневом сегменте, который эту секцию не содержал.

5.3.5. МЕХАНИЗМ ЗАГРУЗКИ ПЕРЕКРЫТИЙ

Существуют два способа загрузки перекрытий в память — автозагрузка и ручная загрузка.

Автозагрузка перекрытий выполняется программой управления перекрытиями автоматически при передаче управления на автозагружаемый сегмент вверх по дереву. Автозагружаемые компоненты в описании перекрытий указываются с помощью индикатора автозагрузки — *.

Индикатор автозагрузки может быть указан для следующих компонентов: имени файла; части описания дерева перекрытий, заключенной в скобки; имени программной секции, содержащей инструкции; имени сегмента; имени ветви, указанной по директиве .FCTR. Индикатор автозагрузки отмечает первый компонент ветви как автозагружаемый. Если описание ветви заключено в скобки, все компоненты ветви являются автозагружаемыми.

При автозагрузке используется механизм загрузки по путям, т. е. вызов из сегмента другого сегмента вверх по дереву обеспечивает автоматическую загрузку в память всех сегментов на пути от вызывающего сегмента до вызванного сегмента.

Указание индикатора автозагрузки (в рассмотренном выше примере для директивы .FCTR) перед скобками, содержащими имена ветвей, обеспечивает автозагрузку всех компонентов в описании дерева перекрытий:

```
ROOT CNTRL — * (AFCTR, BFCTR, C)
```

Автозагрузка сегментов осуществляется за счет создания построителем задач вектора автозагрузки для глобального символа, на который сделана ссылка.

Передача управления на глобальный символ заменяется вызовом подпрограммы автозагрузки $NAUTO. Размещая индикаторы автозагрузки только в тех точках, где загрузка действительно требуется, можно минимизировать число векторов автозагрузки, создаваемых для задачи.

Для программных секций с атрибутом D (данные) вектор автозагрузки не создается. Изменение механизма автозагрузки для сегмента, не содержащего выполняемых инструкций, осуществляется за счет объявления имени сегмента директивой .NAME с атрибутом GBL. При вызове такого сегмента с помощью директивы CALL он загружается автоматически, и управление возвращается в вызывающий сегмент. Далее ссылки на глобальный символ вверх по дереву в программной секции с атрибутом D разрешаются непосредственно.

Ручная загрузка сегментов обеспечивается включением в программу явных вызовов подпрограммы загрузки перекрытий $LOAD. При ручной загрузке загружается только сегмент, указанный в запросе, т.е. загрузка по путям не осуществляется. Ручная загрузка может быть запрошена в синхронном или асинхронном режимах.

При программировании на макроассемблере ручная загрузка осуществляется вызовом подпрограммы $LOAD. Формат макровызова:

```
MOV #PBLK, R0 CALL $LOAD
```

где PBLK — адрес блока параметров, который содержит: длину блока параметров (от 3 до 5 слов); номер флага события, задаваемого для асинхронной загрузки; имя загружаемого сегмента в формате RADIX50; адрес двусловного блока состояния ввода-вывода; адрес подпрограммы асинхронного прерывания по завершении загрузки сегмента.

Ручная загрузка в программах на Фортране осуществляется через подпрограмму MTMLOAD, вызывающую непосредственно подпрограмму $LOAD. Параметры, задаваемые при вызове подпрограммы MNLOAD, аналогичны параметрам в макровызове подпрограммы $LOAD.

5.4. РАЗДЕЛЯЕМЫЕ ОБЛАСТИ

5.4.1. ОБЩИЕ СВЕДЕНИЯ

Разделяемая область представляет собой область оперативной памяти, содержащую коды инструкций или данные, которые могут совместно использоваться любым количеством задач. Разделяемая область, содержащая данные, называется *резидентной общей областью*, а разделяемая

область, содержащая подпрограммы, разделяемые несколькими задачами, — *резидентной библиотекой.*

Разделяемые области обеспечивают:

уменьшение требуемого объема оперативной памяти за счет совместного использования одной копии подпрограммы несколькими задачами;

разделение общих данных между несколькими задачами, осуществляя, таким образом, информационную связь между задачами.

Разделяемая область строится без заголовка и стека, так как она содержит невыполняемый образ. При построении разделяемой области должен быть обязательно создан, кроме файла образа, файл определения символов. Этот файл в последующем используется построителем задач для установления связи между разделяемой областью и задачей, использующей разделяемую область. С целью экономии виртуального адресного пространства (ВАП) задачи, связываемой с разделяемой областью, можно построить разделяемую область со структурой перекрытий, резидентных в памяти. Необходимо отметить, что распределение виртуального адресного пространства для разделяемых областей осуществляется с кратностью 4 Кслова, так как управление памятью связано с использованием аппаратуры диспетчера памяти (ДП).

Разделяемые области могут быть позиционно-независимыми или абсолютными. Позиционно-независимые области могут размещаться в произвольных адресах виртуального адресного пространства задачи. Абсолютные разделяемые области связываются с фиксированными адресами виртуального адресного пространства задачи.

Разделяемая область может быть объявлена позиционно-независимой при следующих условиях:

•область содержит инструкции, которые будут правильно выполняться независимо от их размещения в виртуальном адресном пространстве ссылающейся задачи;

•область содержит данные, которые не являются адресно-зависимыми.

При построении позиционно-независимых областей необходимо учитывать следующие особенности:

•инструкции или данные разделяемой области не должны включаться в неименованную программную секцию (.BLK);

•задача, использующая разделяемую область, не должна иметь программные секции с именами, совпадающими с именами программных секций разделяемой области;

•порядок размещения программных секций (/SQ или /SG) для разделяемой области и ссылающейся к ней задачи должен быть одинаковым.

В файл определения символов для позиционно-независимой разделяемой области включаются определения всех программных секций области. За счет этого задача, связанная с такой разделяемой областью, может ссылаться к ней по имени программной секции.

Для абсолютной разделяемой области в файл определения символов включается только абсолютная секция .ABS. Ссылки к разделяемым инструкциям и данным в этой секции осуществляются по глобальному имени.

5.4.2. СОЗДАНИЕ РАЗДЕЛЯЕМОЙ ОБЛАСТИ

Для создания разделяемой области необходимо построить файл образа задачи и файл определения символов, используемый для связи ссылающейся задачи с разделяемой областью.

Абсолютная разделяемая область строится при указании ключа /—PI, а ключ /PI объявляет разделяемую область как позиционно-независимую. При создании разделяемой области должен указываться всегда ключ /—HD.

В табл. 5.2 приведены различные варианты использования ключей /PI, /CO, /LI и их действие на файл распределения памяти.

Ниже приводится пример создания абсолютной резидентной библиотеки LIBR из файлов A, B, C.

```
>TKB
TKB>LB: |1, 1]LIBR/—HD/MM, LP:, LB: [1, 1]LIBR = A, B, C
TKB>/
      ENTER OPTIONS:
TKB>STACK=0
TKB>PAR = LIBR: 160000: 1000
```

Таблица 5.2

Ключи, указанные совместно с /-HD	Разделяемая область		Имя программной секции	Размещение программных секций в файле определения символов	Размещение символов в файле распределения символов
	абсолютная	позиционно-независимая			
/PI/LI		Да	Совпадает с именем корневого сегмента библиотеки	Одна перемещаемая секция	Все символы со значениями относительно начала секции
/PI/CO		Да	Включаются имена всех программных секций	Все программные секции перемещаемые	Все символы и программные секции
/—PI/LI или /LI	Да		.ABS	Одна абсолютная секция	Все символы абсолютные
/—PI/CO или /CO	Да		Включаются имена всех программных секций	Все программные секции абсолютные	Все символы абсолютные
/PI		Да	Соответствует /PI/CO		
/—PI	Да		Соответствует /—PI/LI		

Область LIBR является абсолютной, поэтому во всех ссылающихся на нее задачах данная область будет отображаться с виртуального адреса 160000.

После создания разделяемой области задачи могут связываться с ней. До установки и запуска задачи разделяемая область должна быть размещена в памяти с помощью команды оператора INS.

5.4.3. РАЗДЕЛЯЕМЫЕ ОБЛАСТИ С ПЕРЕКРЫТИЯМИ, РЕЗИДЕНТНЫМИ В ПАМЯТИ

Структура разделяемой области с перекрытиями, резидентными в памяти, определяется через файл описания перекрытий. При построении разделяемой области ТКВ не включает в файл образа области базовые данные перекрытий (описатели сегментов, векторы автозагрузки) и программы управления перекрытиями. Эти данные включаются в файл определения символов разделяемой области, который используется для связи со ссылающейся задачей.

При построении задачи в ее корневой сегмент включаются базовые данные и глобальные ссылки к программам управления перекрытиями.

Для подавления запросов на загрузку перекрытий каждый сегмент перекрытия разделяемой области создается с атрибутом NODSK.

С помощью необязательного параметра GBLREF можно включить глобальные ссылки в корневой сегмент разделяемой области. В этом случае необходимые векторы автозагрузки и определения генерируются без включения таких ссылок в объектный модуль.

Процедуру создания разделяемой области с резидентными перекрытиями можно представить следующим образом:

•определить структуру перекрытий, содержащую только резидентные в памяти перекрытия;

•включить параметр GBLREF или задать в корневом сегменте модуль, содержащий глобальные ссылки для определения точек входа тех сегментов перекрытий, для которых будут создаваться векторы автозагрузки и глобальные определения.

Ниже приводится пример создания разделяемой области с резидентными в памяти перекрытиями, описываемой следующим образом:

```
.ROOT A—! (∗B, C—∗D)
.NAME   A
.END
```

Корневой сегмент A является пустым и имеет нулевую длину. Все выполняемые инструкции находятся в сегментах, резидентных в памяти, которые создаются из объектных файлов B, C, D, содержащих одноименные глобальные точки входа.

При построении разделяемой области в качестве вводного файла для ТКВ указывается файл, содержащий описание структуры перекрытий разделяемой области:

```
TKB>A/— HD/MM, LP:, SY: A = A/MP
        ENTER OPTIONS:
```

```
TKB>GBLREF = B, C, D
TKB>PAR=A: 160000: 20000
TKB>STACK = 0
TKB>//
```

При построении разделяемой области имя раздела, имя файла образа области и имя файла определения символов должны быть одинаковыми.

В приведенном примере глобальные ссылки на точки B, C и D, включаемые в корневой сегмент, генерируют в файле определения символов для точки C соответствующее определение, а для точек B и D —векторы автозагрузки.

Файл определения символов также содержит базовые данные, необходимые системе управления перекрытиями, для загрузки перекрытий. Эти базовые данные передаются через файл определения символов задаче, ссылающейся к разделяемой области, поэтому загрузка резидентных в памяти перекрытий для разделяемой области не отличается от загрузки перекрытий, резидентных в памяти, создаваемых непосредственно как образ задачи.

Необходимо учитывать следующие ограничения для разделяемых областей с резидентными в памяти перекрытиями:

•разделяемая область не может использовать метод автозагрузки для ссылок на другие резидентные в памяти перекрытия, если имена сегментов не совпадают, другие перекрытия могут загружаться методом ручной загрузки;

•задача не может ссылаться к именованной программной секции, находящейся в сегменте перекрытия разделяемой области. Для возможности таких ссылок программная секция должна быть включена в корневой сегмент разделяемой области;

•в отличие от перекрытий, резидентных в памяти задачи, число векторов загрузки внутри каждой ссылающейся задачи к разделяемой области не зависит от количества действительных ссылок на точки входа и может быть больше количества ссылок.

5.4.4. ГРУППОВЫЕ БИБЛИОТЕКИ

Термин «групповые библиотеки» обозначает одновременно функцию ТКВ и структуру, которую он создает. Групповые библиотеки позволяют динамически отображать разделяемые области, резидентные в памяти. Например, задача может использовать три библиотеки LIB1, LIB2, LIB3 по 16 Кбайт, отображая их через общее адресное окно. Таким образом, разделяемые библиотеки при выполнении задачи используют не 48 Кбайт, а разделят 16 Кбайт виртуального адресного пространства задачи.

Функция групповых библиотек реализуется ТКВ в два этапа. Первый этап обеспечивает установление связей между библиотеками (в случае, если в одной библиотеке имеются ссылки к другой библиотеке группы). Этот этап не зависит от механизма переотображения. Второй этап с использованием директив управления памятью обеспечивает отображение соответствующих групповых библиотек.

Задача связывается с групповыми библиотеками, с перекрытиями, резидентными в памяти, с использованием необязательного параметра CLSTR.

5.4.5. ОБЩИЕ ПРАВИЛА ПОСТРОЕНИЯ ГРУППОВЫХ БИБЛИОТЕК

При построении разделяемых групповых библиотек необходимо выполнять следующие правила:

1. Все библиотеки, кроме первой, должны иметь резидентные перекрытия. Под первой библиотекой понимается первая именованная библиотека, заданная в необязательном параметре CLSTR. Первая библиотека может быть как односегментной, так и с перекрытиями. Все библиотеки с перекрытиями, кроме первой, должны иметь пустой корневой сегмент.

2. Ссылки между групповыми библиотеками разрешаются косвенно через механизм векторных вызовов. Векторный механизм реализуется в ссылающейся задаче и обеспечивает автозагрузку требуемого перекрытия с передачей управления на точку входа.

3. Переопределенные точки входов не должны появляться в файле определения символов последующих библиотек. Это может быть достигнуто исключением каждого переопределенного символа точки входа с помощью необязательного параметра GBLXCL.

4. Для вызываемых библиотечных процедур не должна использоваться передача параметров через стек. Это правило относится ко всем групповым библиотекам, кроме первой, заданной в необязательном параметре CLSTR.

5. Все групповые библиотеки должны быть позиционно-независимыми или построены с одинаковым базовым виртуальным адресом.

6. Передача управления по асинхронным и синхронным системным прерываниям в групповые библиотечные программы недопустима. Это связано с тем, что в момент прерывания требуемая библиотека может оказаться неотображенной.

5.4.6. БИБЛИОТЕКА РЕЖИМА СУПЕРВИЗОРА

Библиотека режима супервизора является резидентной библиотекой, которая используется только в режиме супервизора и поэтому может поддерживаться только в I/D-системах (т. е. только на ВК типа СМ1425).

При использовании обычной задачей библиотеки режима супервизора виртуальное адресное пространство задачи может быть увеличено до 64 Кслов. Это обеспечивается за счет того, что при вызове библиотечной программы осуществляется переключение процессора в режим супервизора. Управляющая программа копирует регистры APR I-пространства режима пользователя в регистры APR D-пространства режима супервизора, а библиотека режима супервизора отображается при этом через APR I-пространства режима супервизора. Таким образом, задача имеет доступ к 32 Кословам своей памяти и 32 Кословам области библиотечных программ. Объем возможного ЛАП задачи увеличивается до 64 Кслов.

Когда с библиотекой режима супервизора компонуются I/D-задачи, в регистры APR D-пространства режима супервизора копируются регистры D-пространства режима пользователя. Таким образом, библиотечные программы режима супервизора могут иметь доступ к пространству данных режима супервизора и к пространству инструкций режима супервизора. Виртуальное адресное пространство I/D-задачи, использующей библиотеку режима супервизора, может быть увеличено до 96 Кслов.

Включение в библиотеку режима супервизора директивы MSDS$ позволяет библиотеке режима супервизора отображать данные в пределах I-пространства режима супервизора, используя APR D-пространства режима супервизора. Таким образом, обеспечивается возможность включить данные в библиотеку режима супервизора.

На содержание библиотек режима супервизора накладываются следующие ограничения:

библиотека может содержать только подпрограммы, вызов которых осуществляется инструкцией JSR;

библиотека не должна содержать подпрограммы, использующиеся для передачи параметров стеков;

если библиотека и вызывающая задача компонуются с библиотечной подпрограммой из системной библиотеки (SYSLIB), то имя точки входа этой подпрограммы должно быть исключено из файла определения символов библиотеки режима супервизора;

если в библиотеку не включена директива MSDS$, то библиотека не должна содержать никаких данных (директивы могут использоваться только в форме — $S).

Для построения и обращения к библиотекам режима супервизора используются необязательные параметры CMPRT, RESSUP (SUPLIB), GBLXCL. Построение библиотеки режима супервизора и компоновка с ней осуществляются так же, как и для резидентной библиотеки.

5.5. КОМАНДНАЯ СТРОКА ДЛЯ ПОСТРОИТЕЛЯ ЗАДАЧ

Командная строка для построителя задач может задаваться двумя способами — с помощью командной строки ТКВ в MCR или команды LINK командного языка DCL.

5.5.1. КОМАНДНАЯ СТРОКА ТКВ

Командная строка ТКВ в общем виде содержит спецификации выводных файлов, отделенных знаком равенства от спецификаций вводных файлов. В командной строке ТКВ можно указать до трех выводных файлов и любое число вводных.

Список выводных файлов может содержать спецификации трех файлов в следующем порядке: файл образа задачи, файл распределения памяти, файл определения символов. Любой выводной файл может быть опущен, и тогда вместо него в командной строке должна быть указана запятая.

Файл образа задачи содержит образ задачи, который загружается в память и запускается на выполнение.

Файл распределения памяти является символьным файлом, содержащим информацию о распределении памяти задачи и разрешении глобальных символов и при необходимости список перекрестных глобальных ссылок.

Файл определения символов является двоичным файлом, содержащим определения глобальных символов в задаче и их виртуальные и перемещаемые адреса. Файл определения используется как вводный файл при создании разделяемых резидентных областей.

Командная строка ТКВ может быть представлена различными способами:

выводные файлы = вводные файлы;

= вводные файлы;

@ косвенный файл.

Здесь каждый файл командной строки задается его спецификацией.

Командная строка ТКВ может задаваться в виде одной строки или в многостроковом формате при использовании большого числа вводных файлов.

Необязательные параметры используются для указания характеристик строящейся задачи.

Признаком для завершения ввода командной строки является знак «//», если нужно завершить работу ТКВ, или знак «/», если предполагается использовать ТКВ для построения другой задачи.

Знак «/», введенный в начале строки за спецификацией вводных файлов, является указанием ТКВ ввести необязательные параметры.

Спецификация вводных и выводных файлов соответствует стандартным системным соглашениям ОСРВМ и содержит устройство, код идентификации, определяющий каталог, имя файла, тип файла, номер версии и любое число ключей. В общем виде спецификация файла имеет формат:

устройство: [гр, чл] имя файла, тип файла; версия/ключ

Компоненты спецификации — устройство, каталог, тип, версия и ключи — являются необязательными. Значения этих компонентов по умолчанию приведены ниже.

Компоненты спецификации файла	**Значение**
устройство:	Имя физического устройства, на котором смонтирован том, содержащий файл. По умолчанию используется устройство, указанное в предыдущей спецификации. Если устройство ни в одной предыдущей спецификации не указано, выбирается SY0:
[гр, чл]	Каталог, задаваемый номером группы и номером члена в группе. Используется для нахождения каталога пользователя (UFD). По умолчанию используется код идентификации, указанный в предыдущей спецификации, или UIC терминала, если UIC не был указан ни в одной предыдущей спецификации
тип	TSK — для файла образа задачи; STB — для файла определения символов; MAP — для файла распределения памяти; OLB —для библиотеки объектных модулей; ODL — для описания перекрытий; CMD — для косвенного командного файла
версия	Для вводного файла — версия с наибольшим номером из всех существующих файлов с указанным именем и типом; для выводного файла — версия на единицу больше максимальной

П р и м е р командной строки ТКВ для построения задачи с именем PRIM:

```
>TKB
TKB>PRIM, PRIM = IN1, IN2
TKB>IN3, IN4
TKB>/
        ENTER OPTIONS:
TKB>STACK=1000
TKB>PRI=100! COMMON = JRNL; RO
TKB>//
```

В приведенном примере ТКВ создает файл образа задачи PRIM. TSK и файл распределения памяти PRIM.MAP из вводных файлов IN1.OBJ, IN2.OBJ, IN3.OBJ, IN4.OBJ. Для указания характеристик строящейся задачи используются необязательные параметры.

5.5.2. КОМАНДА LINK

Команда LINK в общем виде имеет следующий формат, где спецификации выводных файлов разделяются пробелом от спецификаций вводных файлов:

>LINK/выводные файлы/ключи вводные файлы/ключи

Можно задать до трех спецификаций выводных файлов с командными ключами, определяющими файл образа задачи, файл распределения памяти и файл определения символов. Спецификации вводных файлов с вводными ключами разделяются запятыми. Часть ключей может использоваться в качестве командных и вводных, остальные — только как вводные. Если спецификации выводных файлов в команде LINK не задаются, то создается по умолчанию файл образа задачи с именем, совпадающим с именем первого вводного файла.

Если требуется указание большого числа вводных файлов или ключей, команда LINK может задаваться в многостроковом режиме с использованием знака «дефис» как признака продолжения командной строки.

Для задания необязательных параметров в команде LINK необходимо использовать ключ /OPT. В команде LINK, как и в ТКВ, командная строка и необязательные параметры могут передаваться через механизм косвенных командных файлов.

П р и м е р команды LINK для построения задачи с именем PRIM:

```
>LINK/TASK: PRIM/MAP: PRIM/OPTION IN1, IN2,—
→ IN3, IN4
OPTION? STACK=1000
OPTION? PRI = 100! COMMON = JRNL: RO
OPTION?
>
```

Для команды LINK признаком завершения ввода является управляющий символ <CR> (возврат каретки).

5.6. КЛЮЧИ ПОСТРОИТЕЛЯ ЗАДАЧ

В командной строке ТКВ или в команде LINK для определения основных возможностей задачи, а также для управления построением задачи используются ключи. Если ключи не задаются, то ТКВ использует их значения по умолчанию. Ключ задается с помощью символьного кода (2—4 символа) с предшествующим знаком «/». Если ключ начинается со знака минус («—») или с символов NO, действие ключа, определенного следующими символами, отвергается. Например, ключ /CP может иметь одно из трех значений: /CP, /—CP, /NOCP. Каждый ключ может использоваться только с определенными типами вводных или выводных файлов. В команде LINK негативная форма задается только с помощью символов NO.

В табл. 5.3 приведены ключи построителя задач, их описание и значение по умолчанию, а также типы файлов, с которыми эти ключи могут быть заданы: T — файл образа задачи, M — файл распределения памяти, S — файл определения символов, I — вводный файл. Формат ключей и их значения по умолчанию задаются для командной строки ТКВ и для команды LINK. Некоторые ключи не имеют отрицательной формы, а значением по умолчанию может быть положительная или отрицательная формы или отсутствие умолчания.

Таблица 5.3

Формат	Значение	Тип файла	Значение по умолчанию
/AC [:N] /ANS (:N]	Задача является вспомогательным управляющим процессором. Такая задача является привилегированной, N указывает первую пару регистров диспетчера памяти для отображения задачи и может принимать значения 0, 4, 5	T	/—AC N = 5
/AL /CHE: TAS	Задача строится как выгружаемая в область, выделенную в файле образа задачи	T	/—AL /NOCHE: TAS
/CC /CON	Вводной файл является конкатенацией объектных модулей, которые включаются в образ задачи. При отрицании ключа ТКВ включает в образ задачи только первый модуль файла	I	/CC /CON
/CM /COM	Резидентные в памяти сегменты перекрытий выравниваются на границу 256-словного блока для совместимости с возможностями директив управления памяти	T	/—CM /NOCOM
/CP /CHE: SYS	Задача строится как выгружаемая в системный файл выгрузки. Ключи /CP и /AL взаимно исключающие	T	/—CP /NOCHE: SYS
/CR /CRO	Список глобальных перекрестных ссылок добавляется к файлу распределения памяти	M	/—CR /NOCRO

Продолжение

Формат	Значение	Тип файла	Значение по умолчанию
/CO /SHA: COM	Строится разделяемая область. Описание всех программных секций включается в файл распределения памяти	T, S	/CO
/DA /DEB[: файл]	Включить в задачу средства отладки. Если указан ключ с файлом образа задачи или ключ /DEB без имени файла, включается стандартный отладчик LB0: [1, 1] ODT. OBJ. Если указан ключ /DA с вводным файлом или ключ /DEB с именем вводного файла, то вводной файл является отладочным средством	T, I	/—DA
/DL /DEF	Указанный с ключом вводной файл является библиотекой объектных модулей и заменяет системную библиотеку объектных модулей LB0: [1, 1] SYSLIB. OLB для разрешения неопределенных глобальных ссылок	I	/—DL
/EL	Задается размер библиотеки в соответствии с размером, заданным в необязательном параметре PAR	T	/—EL
/FP /COD: FPP	Задача использует процессор с плавающей запятой	T	/FP /COD: FPP
/FU /FUL	Разрешение неопределенных глобальных ссылок с помощью библиотеки объектных модулей, выбираемой по умолчанию, осуществляется для сегментов всех поддеревьев перекрытий. При отрицании ключа разрешаются только ссылки из главного корневого сегмента и текущего дерева	T	/—FU
/HD /HEA	Заголовок задачи включается в образ задачи. При создании разделяемой области необходимо задавать отрицание ключа	T, S	/HD /HEA
/ID /COD: DAT	Задача строится с использованием I/D-пространств	T	/—ID
/IP /IO	Отрицательное значение ключа указывает, что для привилегированной задачи не требуется отображение на внешнюю страницу памяти	T	/IP /IO
/LB /LIB /INC	Вводной файл является библиотечным. Если ключ используется без аргументов, то файл просматривается для разрешения неопределенных глобальных символов. Ключ с аргументами указывает имена модулей, задаваемых как аргументы, которые должны быть включены в образ задачи	I	/—LB
/LI /SHA: LIB	Образ задачи является разделяемой библиотекой, ключ /LI должен указываться вместе с ключом /—HD	T, S	/-LI
/MA /GLO	В файл распределения памяти включается информация о вводном файле. По умолчанию значением ключа для вводного файла является /MA, для файла распределения памяти — /—MA	I, M	/MA /—MA /GLO
/MM /MEM	Целевая система, в которой будут выполняться задачи, имеет диспетчер памяти (ДП). По умолчанию значение ключа устанавливается в зависимости от того, в какой системе строится задача—с ДП или без него	T	/MM /—MM
/MP /OVER	Вводной файл является файлом описания перекрытий для строящейся задачи. Этот файл должен быть единственным вводным файлом в командной строке для ТКВ	I	/—MP
/MU /SHA: TAS	Задача строится как многопользовательская	T	/—MU
/NM /NOWAR	Запрещается печать диагностических сообщений о числе неопределенных символов в сегментах задачи и многократном определении программных секций	T	/—MM /WAR
/OPT	Используется в команде LINK для задания ввода необязательных параметров	T	
/PI /CODE: PIC	Разделяемая область содержит только позиционно-независимые коды	T, S	/—PI /NOCOD:PIC
/PM /POST	При аварийном завершении автоматически распечатывается содержимое памяти задачи	T	/—PM
/PR [:N] /PRIV [:N]	Задача строится как привилегированная, N принимает значение 0, 4, 5 и определяет первую пару регистров диспетчера памяти, используемых для отображения задачи	T	/—PR
/RO /RES	Разрешается распознавание в файле описания перекрытий оператора резидентных в памяти перекрытий —«!». При отрицательном значении ключа проверяется корректность использования этого оператора, но перекрытия строятся как диск-резидентные	T	/RO /RES
/SE /REC	Сообщения могут посылаться задаче из другой задачи с помощью системной директивы SEND$	T	/SE /REC
/SG /SEG	Программные секции группируются с кодами доступа (сначала RW, затем RO) и размещаются внутри групп в алфавитном порядке имен секций	T	/—SG /NOSEG

Продолжение

Формат	Значение	Тип файла	Значение по умолчанию
/SH /MAP[:файл]	Создается короткая форма файла распределения памяти	M	/SH
/SL /SLA	Задача строится как подчиненная по отношению к инициирующей задаче	T	/—SL
/SP /PRIN	Для вывода файла распределения памяти запрашивается системный вывод	M	/SP /PRIN
/SQ /SEQ	Программные секции группируются в соответствии с кодами доступа (сначала RW, затем RO) и размещаются внутри групп в порядке их ввода	T	/—SQ
/SS /SEL	В таблицу глобальных символов включаются только определения глобальных символов, на которые имеются ссылки из других вводные файлов	I	/—SS
/SYM	Указывается в команде LINK для создания файла определения символов	S	
/ТАБ[:файл]	Указывается в команде LINK для создания файла образа задачи	T	/TAS
/TR /TRA	Задача строится с установленным битом слежения T в PSW. После каждой инструкции будет возникать синхронное системное прерывание по биту слежения	T	/—TR
/WI /WID	Указывает широкий формат (132 символа) распечатки файла распределения памяти. При отрицании ключа ширина формата устанавливается равной 80 символам	M	/WI /WID
/XH /EXT	Задача строится с заголовком задачи, размещаемым при ее выполнении вне системной динамической памяти	T	/XH /EXT
/XT[:N] /ERR[:N]	ТКВ завершает построение задачи после обнаружения N ошибок	T	/—XT N=1

5.7. НЕОБЯЗАТЕЛЬНЫЕ ПАРАМЕТРЫ

Необязательные параметры используются для указания характеристик строящейся задачи. Построитель задач запрашивает ввод необязательных параметров, если за спецификацией вводных файлов в командной строке ТКВ указывается знак «/» или ключ /OPTION в команде LINK. Ф о р м а т необязательного параметра:

ключевое слово = список аргументов

Аргументы в списке разделяются знаком двоеточие (:). В одной строке может быть задано несколько необязательных параметров, разделенных знаком «!».

Некоторые параметры имеют несколько списков аргументов, для разделения этих списков используется запятая.

П р и м е р :

```
TASK = NAME! PRI=100
COMMON = DATA: RW! LIBR: RO
```

Далее приводится формат необязательных параметров и их назначение.

Параметр ABORT обеспечивает перезапуск ТКВ без создания вводных файлов. Используется при ошибке, допущенной ранее, в последовательности ввода командных строк или необязательных параметров. Ф о р м а т :

ABORT = N

где N — любое число, значение которого игнорируется.

Параметр ABSPAT обеспечивает до восьми корректировок указанного сегмента. Ф о р м а т :

ABSPAT = имя сегмента: адрес: значение 1: значение 2: ...

где имя сегмента — имя сегмента, содержит от 1 до 6 символов;

адрес—восьмеричный адрес первой корректировки;

значение 1, 2, ... — восьмеричные числа в пределах от 0 до 177777, записываемые в последовательные ячейки, относительно заданного адреса.

Параметр ACTFIL объявляет число файлов, которое задача может иметь одновременно открытыми. Для каждого открытого файла выделяется 512 байт памяти. Параметр используется только для программ на Фортране. Ф о р м а т :

ACTFIL = N

где N — число файлов. Значение по умолчанию: ACTFIL=4.

Параметр ASG назначает допустимые логические номера для одного или нескольких устройств. Ф о р м а т :

ASG = имя устройства: номер 1: номер 2: ... номер 8

Значение по умолчанию:

ASG = SY0: 1: 2: 3: 4: TI0: 5: CL0: 6

Параметр CLSTR объявляет группу системных разделяемых областей (от двух до шести), с которыми связываются задачи, и имеет к ним доступ с использованием общего разделяемого виртуального адресного пространства. Разделяемые общие области данных или разделяемые библиотеки и соответствующие им файлы определения символов должны быть размещены на устройстве LB: в каталоге [1,1]. Ф о р м а т :

CLSTR = имя 1, имя 2, ..., имя N: ключ [:регистр]

где имя 1, имя 2, ... — имена разделяемых областей. В группе может быть задано от двух до шести областей. Первая разделяемая область отображается при активизации задачи, другие области переотображаются в общее разделяемое виртуальное адресное пространство при обращении к ним;

ключ — определяет общий режим доступа ко всем разделяемым областям группы: RW — чтение и запись, RO — только чтение;

регистр — определяет номер (от 1 до 7) первой пары регистров диспетчера памяти, используемых для отображения разделяемых областей, и указывается только для позиционно-независимых областей.

Параметр CMPRT определяет точку входа в программу завершения, которая переключает процессор из режима супервизора в режим пользователя и возвращает управление пользовательской задаче после ее работы с библиотекой режима супервизора. Ф о р м а т :

CMPRT = имя

где имя — точка входа.

Параметры COMMON и LIBR являются идентичными и объявляют, что задача использует системную разделяемую область (соответственно системную разделяемую область данных или системную разделяемую библиотеку). Ф о р м а т :

COMMON = имя: код доступа [:регистр]
LIBR= имя: код доступа [: регистр]

где имя — имя системной разделяемой области; файл определения символов разделяемой области должен находиться на устройстве в каталоге [1, 1];

код доступа — RW или RO;

регистр — определяет номер (от 1 до 7) первой пары регистров ДП для отображения разделяемой области. Он указывается только для позиционно-независимых областей.

Параметр DSPPAT обеспечивает до восьми корректировок D-пространства в I/D-задачах для указанного сегмента. Ф о р м а т :

DSPPAT = имя сегмента: адрес: значение 1: значение 2: ...

где значение аргументов идентичны аргументам параметра ABSPAT.

Параметр EXTSCT объявляет расширение размера программной секции. Для программной секции с атрибутом CON секция расширяется на указанное число байтов. Для секции с атрибутом OVR размер программной секции устанавливается равным длине расширения, если длина расширения больше длины секции. Ф о р м а т :

EXTSCT = имя программной секции: N

где N — восьмеричное число, определяющее расширение программной секции в байтах.

Параметр EXTTSK объявляет выделение дополнительной памяти для задачи, устанавливаемой в системно-управляемом разделе. Для I/D-задач расширяется только D-пространство. Ф о р м а т :

EXTTSK=N

где N — десятичное число слов приращения задачи.

Параметр FMTBUF определяет размер рабочего буфера, выделяемого для компиляции спецификаций формата, вводимых во время выполнения программы. Параметр используется только для программ на Фортране. Ф о р м а т :

FMTBUF=N

где N—десятичное число, определяемое количество символов самой длинной спецификации формата. По умолчанию N =132.

Параметр GBLDEF объявляет определение глобального символа, отвергая определение этого символа во всех модулях. Ф о р м а т :

GBLDEF=имя глобального символа: N

где N — восьмеричное число в пределах от 0 до 177777.

Параметр GBLINC обеспечивает включение определения одного или нескольких глобальных символов, заданных в параметре, в файл определения символов. Ф о р м а т :

GBLINC = имя 1, имя 2, ..., имя N

где имя 1, ..., имя N — имена включаемых глобальных символов.

Параметр GBLPAT обеспечивает возможность 1—8 корректировок относительного глобального символа. Ф о р м а т :

GBLPAT=имя сегмента: символ [+ /—смещение]: N1: N2: ... N8

где смещение — восьмеричное число, определяющее смещение относительно глобального символа;

N1, N2, ..., N8 — восьмеричные числа, записываемые последовательно в слова с заданным смещением относительно глобального символа.

Параметр GBLREF объявляет ссылку на глобальный символ. Ссылка порождается в корневом сегменте. Ф о р м а т :

GBLREF=имя глобального символа

Параметр GBLXCL обеспечивает возможность исключения указанных глобальных символов из файла определения символов. Ф о р м а т :

GBLXCL = символ 1, символ 2, ...

Параметр MAXBUF объявляет максимальный размер буфера записей для файлов, обрабатываемых задачей. Параметр используется только для программ на Фортране. Ф о р м а т :

MAXBUF = N

где N — десятичное число, ОПРЕДЕЛЯЮЩЕЕ максимальный размер записи в блоках.

Параметр ODTV объявляет указанный глобальный символ адресом таблицы векторов SST-прерываний программы отладчика. Глобальный символ должен быть определен в корневом сегменте. Ф о р м а т :

ODTV=символ: N

где N — длина таблицы векторов SST-прерываний (в словах).

Параметр PAR определяет раздел, для которого строится задача. Задачу можно установить в любом разделе, если его размеры позволяют разместить задачу. Значением по умолчанию является PAR = GEN. Ф о р м а т :

PAR = имя раздела [:база:длина]

где база — восьмеричный базовый адрес раздела; для задач он должен быть нулевой, а для разделяемых областей — кратным 4 Кловам;

длина — восьмеричное число байтов, выделяемых задаче в разделе. Построитель задач автоматически в пределах раздела расширяет размер задачи до этого значения, независимо от фактического объема памяти, требуемого задачей.

Параметр PRI определяет приоритет задачи. В системе с многопользовательской защитой задача, запускаемая с непривилегированного терминала, не может иметь приоритет выше 50. Ф о р м а т :

PRI = приоритет

где приоритет — десятичное число от 1 до 250.

Параметры RESCOM и RESLIB являются идентичными и объявляют, что задача использует резидентную разделяемую область (соответственно резидентную разделяемую область данных или резидентную библиотеку). Ф о р м а т :

RESCOM=файл/код доступа [:регистр]
RESLIB = файл/код доступа [: регистр]

где файл — спецификация файла образа резидентной области данных или резидентной библиотеки; файл определения символов должен находиться в том же каталоге, в каком находится файл образа;

код доступа — RW или RO;

регистр — определяет номер (от 1 до 7) первой пары регистров ДП для отображения разделяемой резидентной области и может указываться только для позиционно-независимых разделяемых областей.

Параметр RESSUP объявляет, что задача использует пользовательскую библиотеку режима супервизора. Ф о р м а т :

RESSUP = файл/[—] SV [: регистр]

где файл — спецификация файла образа библиотеки режима супервизора; файл определения символов должен находиться в том же каталоге, в каком расположен образ библиотеки;

/[—]SV — ключ /SV или /—SV указывает, должны ли системные векторы переключения режимов процессора включаться в пользовательскую задачу;

регистр — определяет номер (от 0 до 7) первой пары регистров диспетчера памяти режима супервизора для отображения библиотеки режима супервизора. Указывается только для позиционно-независимых библиотек.

Параметр ROPAR объявляет раздел, в котором будет размещаться часть многопользовательской задачи с режимом доступа «только чтение» (RO). Ф о р м а т :

ROPAR = имя раздела

Параметр SUPLIB объявляет, что задача использует системную библиотеку режима супервизора. Ф о р м а т :

SUPLIB = имя:/[—]SV [: регистр]

где имя — имя системной библиотеки режима супервизора. Библиотека и ее файл определения символов должны находиться на устройстве LBO: в каталоге [1, 1];

/[—]SV —ключ /SV или /—SV указывает, должны ли системные векторы переключения режимов процессора включаться в пользовательскую задачу;

регистр — определяет номер (от 0 до 7) первой пары регистров ДП режима супервизора для отображения позиционно-независимых библиотек режима супервизора.

Параметр TASK определяет имя задачи. Ф о р м а т :

TASK = имя задачи

Параметр TSKV объявляет глобальный символ адресом таблицы векторов SST-прерываний задачи. Указанный символ должен быть определен в главном корневом сегменте. Ф о р м а т :

TSKV = символ: N

где N — длина таблицы векторов SST прерываний в словах.

Параметр UIC объявляет код идентификации задачи при ее выполнении. Ф о р м а т :

UIC=[гр, чл]

Значением по умолчанию является код идентификации, под которым выполняется ТКВ (обычно UIC терминала).

Параметр UNITS объявляет число логических номеров устройств, используемых задачей. Ф о р м а т :

UNITS = N

где N — число в пределах от 0 до 250, указывающее максимальное число логических номеров устройств, используемых задачей. Значение по умолчанию—UNITS =6.

Параметр VSECT определяет распределение виртуальной памяти для именованной программной секции, что позволяет создавать большие структуры данных и ссылаться к ним с помощью директив управления памятью. Ф о р м а т :

VSECT=имя секции: база: окно [: район]

где база — виртуальный базовый адрес программной секции, кратный 4 Ксловам;

окно — размер виртуального адресного пространства, выделяемого программной секции. Сумма величины базы и размера окна не должна превышать значение 177777;

район — размер физической памяти (в 32-словных блоках), выделяемой в районе задачи для размещения программной секции. Секция может отображаться в эту физическую память при выполнении задачи. Параметр «район» является необязательным, при его умолчании предполагается, что район для отображения программной секции будет создаваться при выполнении задачи по директиве GRRG$.

Параметр WNDWS объявляет дополнительное число адресных окон, необходимых задаче, кроме окон, используемых для отображения образа задачи и разделяемых областей. Ф о р м а т :

WNDWS = N

где N — число от 1 до 23. Значением по умолчанию является WNDWS = 0.

6. Программные средства контроля и управления системой

6.1. ПРОГРАММА КОНСОЛЬНОГО ПРОТОКОЛИРОВАНИЯ (COLOG)

Программа консольного протоколирования (COLOG) предназначена для управления запросами ввода-вывода на консоль (устройство CO:) и записи системных сообщений на терминал и/или в файл протокола с указанием времени их поступления. Программа (COLOG) включается в ОСРВМ во время генерации.

Когда процесс консольного протоколирования активен, ввод-вывод на устройство CO: контролируется задачей вывода на консоль (COT...) и консольным драйвером (CODRV). Когда программа COLOG не активна, CO: является псевдоустройством, и операции ввода-вывода на устройство CO: передаются терминальному драйверу. В обоих случаях вывод на устройство CO: может быть направлен на терминал, который назначен консольным, по команде REDIRECT. Однако лучше использовать команду SET/COLOG.

Использование программы COLOG для управления запросами ввода-вывода на CO: имеет два преимущества:

консольное протоколирование позволяет посылать вывод сообщений на CO: на консольный терминал и/или в файл протокола. Эта гибкость дает возможность просмотреть файл протокола в любое время и освободить терминал, когда нет необходимости в немедленном выводе на консоль;

программа COLOG отмечает время вывода сообщения на CO: в следующем формате:

нн:мм:сс (часы, минуты, секунды)

Кроме того, после изменения даты перед выводом первого сообщения в протокол выводятся текущее время и дата.

Пользователь может использовать задачу COT... для взаимодействия с программой COLOG.

COT... является привилегированной задачей. С привилегированного терминала можно выводить команды для запуска и прекращения процесса консольного протоколирования, назначения консольного терминала и задания файла протокола. Программа поддерживает следующие команды:

SET/COLOG — сообщить текущие назначения консольного терминала и файла протокола;
SET/COLOG = ON— запустить процесс консольного протоколирования;
SET/COLOG = OFF—прекратить процесс консольного протоколирования;
SET/COLOG/NOC[OTERM]—запретить вывод на консольный терминал;
SET/COLOG/COT [ERM] [= TTNN:] — изменить консольный терминал;
SET/COLOG/NOLOGFILE — запретить вывод в файл протокола;
SET/COLOG/LOG [FILE] [= файл]—изменить заданный файл протокола.

Буфер в COT... допускает сообщения размером не более 250(10) байт. Установка размеров буфера для устройства CO: командой SET/BUFFERSIZE не изменяет это ограничение.

В задачах могут использоваться GIO-запросы на ввод данных с консольного терминала. Например, консольный терминал может использоваться для ввода ответа на подсказку системной задачи. COT... не записывает в файл протокола данные, введенные с консольного терминала.

Формат команды программы COLOG:

SET/COLOG [функция]...

где SET— команда SET программы MCR;
COLOG — ключ консольного протоколирования;
функция — одна из функций консольного протоколирования, при задании которых можно использовать сокращение до трех символов.

В командной строке можно использовать составные команды программы COLOG. Во время активного процесса COLOG для определения состояния CO можно применять команду DEV программы MCR, например:

```
>DEV CO:
CO0: LOADED
```

Это означает, что драйвер CODRV загружен и что задача COT... управляет выводом системных сообщений на CO:.

Текущий файл протокола открыт для записи, если установлена функция /LOGFILE. Этот файл протокола можно прочитать, применяя программу PIP с ключом разделенного чтения (/SR).

В следующих примерах, иллюстрирующих использование программы COLOG, предполагается, что устройство LB: назначено на DP0:

Пример 1.

```
SET/COLOG
COT— —
CONSOLE = NONE
LOGFILE = NONE
```

Строки выводятся на терминал, с которого подана команда назначения текущего консольного терминала (нет) и файл протокола (нет), т. е. программа COLOG неактивна.

Пример 2.

```
>DEV CO:
CO0 TT0:
```

Команда DEV показывает, как переадресован CO:. Программа COLOG неактивна, и терминальный драйвер переадресует устройство CO: на TT0:

Пример 3.

```
>SET/COLOG = ON
```

Запуск COLOG.

Пример 4.

```
>DEV CO:
CO0: LOADED
>  SET/COLOG
COT— —
CONSOLE = TT0:
LOGFILE = DP0: [1, 4]CONSOLE.LOG; 1
```

Программа COLOG активна, так как драйвер CODRV загружен. /COLOG показывает, что задача COT... переадресует CO: на TT0: и что DP0: [1, 4] CONSOLE.LOG; 1—файл протокола.

Пример 5.

```
>SET /COLOG/COTERM = TT5:
```

Консольный терминал назначается на TT5:.

Пример 6.

```
>SET /COLOG/LOGFILE = DK:[301, 55]TEST
```

Файл протокола назначается на DK: [301, 55] TEST. LOG. Задача COT... заполняет недостающие части спецификации файла частями из спецификации по умолчанию.

Пример 7.

```
>  SET/COLOG
COT— —
CONSOLE = TT5:
LOGFILE = DK:[301, 55]TEST.LOG;l
```

Сообщаются назначения текущего консольного терминала (TT5:) и файла протокола (DK: [301, 55] TEST.LOG; 1), установленные в результате действия команд в примерах 5 и 6.

Пример 8.

```
>SET /COLOG/NOCOT
>SET /COLOG
COT— —
CONSOLE=NONE
LOGFILE = DK:[301, 55]TEST.LOG;l
```

Отменяется консольный терминал и сохраняется текущий файл протокола (DK: [301, 55] TEST. LOG; 1), что подтверждается в следующем примере.

Пример 9.

```
>DEV CO:
CO0: LOADED
```

В этом примере показано, что хотя вывод на консольный терминал запрещен по /NOCOTERM, программа COLOG еще активна (CODRV загружен), но устройство CO: не переадресовано на другой терминал.

Пример 10.

```
>SET /COLOG/COTERM/NOLOG
>SET /COLOG
COT— —
CONSOLE = TT5:
LOGFILE = NONE
```

Показана составная команда, по которой восстанавливается самое последнее назначение консольного терминала (TT5:) и запрещается вывод файла протокола.

Пример 11.

```
>SET /COLOG/COTERM = TT1:/LOG =
>SET /COLOG
COT— —
C0NS0LE = TT1;
LOGFILE = DK: [301, 55] TEST.LOG;2
```

Показана составная команда, по которой консольный терминал назначается на TT1: и восстанавливается самый последний файл протокола, но уже с увеличенным на единицу номером версии.

Пример 12.

```
>SET /COLOG = OFF
>SET /COLOG
COT— —
CONSOLE = NONE
LOGFILE=NONE
>DEV CO:
CO0: TT1:
```

Прекращение процесса консольного протоколирования.

Пример 13.

```
>SET /COLOG/COT = TT0:/LOG >SET /COLOG
COT— —
CONSOLE = TT0:
LOGFILE = DP1:[l, 4]CONSOLE.LOG;2
```

Представлена составная команда, по которой консольный терминал назначается на устройство TT0: и восстанавливается файл протокола со спецификацией по умолчанию и увеличенной на единицу версией.

Пример 14.

```
>PIP TI: = LB:  [1, 4] CONSOLE.LOG/SR
 16:43:19   LOGOUT USER [1, 4] TT5:
 16:43:35   LOGIN USER ALPHA [7, 334] TT5:
 16:45:07   LOGIN USER BETA [7, 42]TT1:
 16:46:21   ∗ ∗ ∗ DMO:— —DISMOUNT COMPLETE
 16:46:32   LOGIN USER GAMMA [240, 240] TT3:
 16:47:22   LOGIN USER LEV  [7, 373] TT2:
 16:47:38   ∗ ∗ ∗ LPO:— —NOT READY
 16:47:49   LOGOUT USER  [1, 4]TT5:
 16:47:58   LOGOUT USER  [7, 373] TT2:
 16:48:01   LOGIN USER ALPHA [7, 334]TT5:
 16:49:40   LOGIN USER GAMMA [240, 240] TT0:
```

В команде на чтение открытого файла протокола использован ключ /SR. Каждое сообщение файла начинается с отметки времени.

6.2. ПОДСИСТЕМА РЕКОНФИГУРАЦИИ КОМПЛЕКСА

Подсистема реконфигурации комплекса (ПРК) позволяет управлять ресурсами комплекса путем включения или отключения внешних устройств.

С помощью ПРК возможно изменение набора технических средств, доступных для использования в комплексном режиме. Например, после загрузки системы можно перевести накопитель на магнитных дисках в автономный режим и вместо него включить в текущую конфигурацию другой накопитель.

ПРК состоит из следующих компонентов:

•программы интерфейса с оператором (CON);

•загружаемого драйвера реконфигурации (RD:);

•программы реконфигурации (HRC).

Для программ CON и HRC должно быть зарезервировано достаточное пространство в файле выгрузки и в оперативной памяти для обеспечения их одновременной загрузки. Если одновременная загрузка задач невозможна, команды оператора, адресуемые программой CON, завершаются с ошибкой.

Программа CON получает и обрабатывает команды оператора для реконфигурации системы. После проверки правильности команд программа CON формирует запросы QIO и передает их драйверу реконфигурации RD:

Драйвер реконфигурации RD: обеспечивает интерфейс между задачами пользователя (в том числе и программой CON) и программой HRC. Он служит связующим компонентом между задачами, генерирующими запросы реконфигурации, и компонентами, исполняющими их.

Драйвер RD: получает пакеты QIO от всех задач и размещает их в очередь к программе HRC. Перед их передачей в очередь к HRC драйвер реконфигурации производит проверку допустимости параметров пакета QIO. Такая проверка позволяет убедиться в допустимости кодов функций, диапазона задаваемых адресов и т. д.

Программа HRC осуществляет фактическое выполнение операций по реконфигурации. Для этого программа HRC вызывает различные подпрограммы управляющей программы, которые в свою очередь вызывают соответствующие драйверы устройств. Информационные запросы о конфигурации комплекса программа HRC обслуживает без вызова управляющей программы. При этом информация о конфигурации передается непосредственно в буфер пользователя.

После завершения работы HRC в запрашивающую задачу возвращается информация о текущем состоянии устройств. Это позволяет программе HRC взаимодействовать и с задачами, отличными от программы CON.

Таблицы устройств в ОСРВМ содержат информацию о их состоянии. При изменении конфигурации системы программа HRC модифицирует поля состояния, разрешая или запрещая к ним доступ.

Устройства могут находиться в одном из четырех состояний:

ON LINE — устройство в комплексном состоянии (доступ разрешен);

OFF LINE — устройство в автономном состоянии (доступ запрещен);

устройство отмечено для перехода в ON LINE — устройство перейдет в состояние ON LINE после подключения (доступ запрещен);

устройство отмечено для перехода в OFFLINE—устройство в неопределенном состоянии (доступ запрещен).

Устройство в состоянии ON LINE доступно для использования. Это означает, что между процессором и данным устройством установлена связь и запрос на получение состояния контроллера или устройства был выполнен успешно. Для выполнения операций ввода-вывода на накопителях на магнитных дисках и лентах в ОСРВМ они должны быть смонтированы.

Когда устройство находится в состоянии OFF LINE, доступ к нему запрещен. В случае обращения к такому устройству запрос ввода-вывода завершается с кодом ошибки IE.OFL. В результате перевода в состояние OFF LINE контроллера переходят в состояние «устройство отмечено для перехода в OFF LINE» все устройства, присоединенные к данному контроллеру.

Состояние «устройство отмечено для перехода в ON LINE» означает, что был выполнен запрос о переводе устройства или контроллера в ON LINE, однако связь между ними и процессором содержит компоненты, находящиеся в состоянии OFF LINE. Попытки выполнить операции ввода-вывода приводят к ошибкам с кодом IE. OFL. После установки связи программа HRC переводит такие устройства в состояние ON LINE.

Состояние «устройство отмечено для перехода в OFF LINE» означает, что произошла непредсказуемая ошибка во время попытки перевода указанного устройства в состояние ON LINE. Оператору следует выполнить команду перевода в состояние OFF LINE перед попыткой повторить команду перевода в ON LINE.

Для перевода устройства в состояние ON LINE программа HRC проверяет, загружен ли драйвер (для загружаемых драйверов) и возможен ли доступ контроллера к данному устройству (контроллер должен быть в состоянии ON LINE). Если доступ контроллера к устройству возможен, программа HRC модифицирует соответствующее слово состояния, где отмечается, что устройство находится в состоянии ONLINE. Если контроллер доступа к устройству не имеет, программа HRC устанавливает состояние «устройство отмечено для перехода в ON LINE». После установки связи программа HRC изменит состояние данного устройства на ONLINE.

Перед переводом устройства в состояние OFF LINE программа HRC проверяет, не приведет ли изменение состояния устройства к потере или искажению данных на носителе. Признаком активности устройства является контекст, т. е. условие, которое указывает на разрешение или выполнение операций ввода-вывода.

Условия, определяющие контекст устройства:

- наличие задачи, присоединившей устройство;
- устройство является терминалом, зарегистрированным в системе;
- устройство монтировано;
- для устройства разрешено кэширование.

В состояние OFF LINE нельзя перевести контроллер, к которому присоединены устройства, имеющие контекст.

ПРК принимает команды реконфигурации с терминала оператора и из косвенного командного файла. Командная строка ПРК состоит из двух частей: имени команды и ее параметров.

Интерпретатор DCL не поддерживает программу CON.

При вводе команд для программы CON их имена можно сокращать до трех букв. Например, команда ONLINE может быть задана в следующих форматах:

```
CON>ONL DDNN:
```

или

```
CON>ONLINE DDNN:
```

где DDNN: — имя и номер устройства.

Программа CON может быть запущена в интерактивном режиме двумя способами:

- командной строкой MCR, что позволяет запустить программу CON, выполнить одну команду и вернуть управление MCR.

П р и м е р :

```
MCR>CON команда
```

- путем явного запуска CON и выполнения ряда команд в интерактивном режиме, например:

```
MCR>CON
CON > команда 1
CON > команда 2
      .
      .
      .
CON>CTRL/Z
```

где CTRL/Z вводится для выхода из программы CON.

Кроме того, могут быть использованы косвенные командные файлы, содержащие команды реконфигурации. Это позволяет не занимать терминал пользователя для проведения реконфигурации и автоматизировать необходимые изменения системы. Например, команда

```
MCR> CON@ CONFIG
```

приводит к выполнению команд, содержащихся в файле CONFIG. CMD, и возврату управления в MCR.

Максимальная глубина вложения косвенных командных файлов равна 3.

Ниже приведены команды реконфигурации и их назначение.

Команда	**Назначение**
BUILD	Создает во внутреннем буфере последовательность команд, которая при выполнении восстанавливает текущую конфигурацию системы
CLEAR	Уничтожает во внутреннем буфере последовательность команд, созданную командой BUILD
DISPLAY	Отображает конфигурацию и состояние устройств в данный момент времени

ESTATUS	Выдает текущее состояние заданного устройства
HELP	Выводит справочную информацию о работе CON
IDENT	Выводит номера текущих версий программ CON и HRC, а также дату и время построения
LIST	Распечатывает результат работы команды BUILD
OFFLINE	Присваивает состояние OFF LINE для устройства, находящегося в текущей конфигурации
ONLINE	Выполняет попытку перевести устройство в состояние ON LINE
SET	Изменяет вектор и адрес регистра контроля и состояния устройства

Ключи, используемые в командах программы CON:

/HE — выводится справочная информация;
/NOMSG — подавляется вывод на терминал сообщений об ошибках, выдаваемых программой CON.

Команда BUILD создает во внутреннем буфере последовательность команд, которая при выполнении восстанавливает текущую конфигурацию системы. Формат:

CON>BUI [LD]

Команда CLEAR уничтожает во внутреннем буфере последовательность команд, созданную командой BUILD. При этом CLEAR не уничтожает файл на диске, созданный командой LIST. Формат:

CON>CLEAR

Команда DISPLAY отображает конфигурацию и состояния устройств в данный момент времени. Формат:

CON>DIS [PLAY] [параметр(ы)] [FOR строка]

где параметрами могут быть следующие ключевые слова:
UNI [TS]—распечатать все устройства, находящиеся в данной конфигурации. При этом каждому устройству ставится в соответствие его контроллер;
CON [TROLLERS] — распечатать все контроллеры устройств в данной конфигурации;
FUL [L] — распечатать флаги состояния для каждого устройства и контроллера в текущей конфигурации. Флаги распечатываются с использованием следующих обозначений:
ACC [PATH] — устройство имеет работоспособный канал передачи данных;
CON[TEXT] — для дисков или лент — устройство монтировано, для
терминалов — пользователь зарегистрирован в системе, для других устройств — подсоединен к задаче;
DRI [VER] — для устройства — драйвер загружен. Для контроллера — драйвер имеется в системе;
OFF[LINE] — устройство в состоянии OFF LINE;
ON [LINE] — устройство в состоянии ON LINE (или «отмечен для ON LINE»);
PRV[_DIAG] — (используется совместно с OFFLINE и ONLINE)
устройство отмечено для ON LINE или OFF LINE, однако выполнение перехода задерживается неопределенным состоянием контроллера;
UNK[NOWN]—адрес регистра контроллера и состояния для данного устройства равен 160000 и, следовательно, устройство недоступно;
ALL — распечатывает такую же информацию, как и с использованием ключевого слова FULL. При этом добавляется адрес регистра контроля и состояния, а также вектор прерывания контроллера;
ATT [RIBUTES] — распечатывает адреса вектора прерываний и регистра контроля и состояния контроллеров внешних устройств.

В команде **DISPLAY** допускается использование более одного ключевого слова в одной командной строке. При этом некоторые комбинации ключевых слов могут быть бессмысленными. В этом случае программа CON игнорирует недопустимые ключевые слова.

Если ключевые слова опущены, программа CON выводит логические имена и флаги состояний для всех устройств и контроллеров в текущей конфигурации.

FOR — дополнительный параметр, позволяющий объединить выводимую информацию в группы для устройства или набора устройств. Его применение позволяет задать набор таких устройств. Формат:

FOR строка

Строка задает устройство для отображения. Возможно использование звездочки (*) и имени устройства для распечатки всех устройств, подсоединенных к заданному контроллеру, и двух звездочек (**) и флага для распечатки всех устройств, имеющих такой же флаг.

Пример 1.

CON>DISK FULL FOR DU

Командная строка для отображения всей информации о контроллерах DU.

Пример 2.

CON>DIS FULL FOR ** ONL

Командная строка для распечатки всех устройств, имеющих флаг состояния ON LINE.

Команда ESTATUS приводит к завершению работы программы CON и выдаче слова состояния заданного устройства. Она используется только с процессором косвенных командных файлов (AT.). Полученное слово состояния позволяет получитьинформацию о конфигурации. Значение битов в слове состояния приведено в табл. 6.1.

Таблица 6.1

Номер бита	Значение бита	Описание	Флаг
0	1	Значение кода завершения	
1	2	0 = предупреждение	
2	4	1=успешное завершение	
		2= ошибка	
		3= ошибка разрыва	
6	100	Устройство — вторичный контроллер	
7	200	Устройство-контроллер	
8	400	0 — устройство в состоянии ON LINE	ONLINE
		1 — устройство в состоянии OFF LINE	OFFLINE
9	1000	Устройство выполняет привилегированные или диагностические функции	PRV_DIA
12	10000	Адрес регистра контроля и состояния равен 16000	UNKNOWN
13	20000	Устройство имеет работоспособный канал передачи данных	ACCPATH
14	40000	Устройство имеет контекст	CONTEXT
15	100000	Драйвер устройства загружен	DRIVER

Пример:

```
>@T1                          запуск AT.
AT.>.ENABLE SUBSTITUTION      разрешение режима замены
AT.>CON ESTAT DM1:            ввод команды для задачи CON
>CON ESTAT DM1:               отображение команды
AT.>;'<EXSTAT>'               распечатка <EXSTAT>
>; 120401                     отображение содержимого системой
AT.> ^Z                       выход из AT.
> @<EOF>
>
```

Данный пример иллюстрирует получение состояния завершения программы CON для устройства DM1:, которое запоминается в переменной <EXSTAT>.

Значение полученного кода состояния 120401 расшифровывается с помощью табл. 6.1 следующим образом:

100000 — драйвер устройства загружен;
20000 — устройство имеет работоспособный канал передачи данных;
400 — устройство в состоянии OFFLINE;
1 — код завершения успешный.

Команда HELP предназначена для получения справочной информации о работе программы CON. Формат:

CON>HEL[P]

Команда IDENT предназначена для вывода номеров текущих версий программ CON и HRC, а также времени и даты их построения. Формат:

CON>IDENT

Команда LIST предназначена для распечатки результатов работы команды BUILD на терминал или их вывода в указанный файл. Полученный файл может быть использован для восстановления текущего состояния конфигурации. Если буфер пустой (команда BUILD не была выполнена), программа CON выдает следующее сообщение:

CON— — COMMAND LIST IS EMPTY.NOTHING TO PRINT.

(Список команд отсутствует. Печатать нечего)

Если с помощью команды LIST создать файл на диске, то его в дальнейшем можно выполнить для восстановления сохраненной конфигурации. Ф о р м а т :

CON > LIST [спец. файла]

где спец. файла — спецификация файла в стандартном формате ОСРВМ.

П р и м е р :

```
CON>BUILD
CON>LIST
ONLINE DU0:
ONLINE DU1:
ONLINE MS0:
ONLINE MS1:
ONLINE CO0:
ONLINE TT0:
ONLINE TT1:
ONLINE TT2:
ONLINE TT3:
ONLINE DK0:
ONLINE DK1:
ONLINE LP0:
ONLINE NL0:
ONLINE DM0:
ONLINE DM1:
```

Команда BUILD создает последовательность команд, отражающих текущую конфигурацию системы. С помощью команды LIST состояние конфигурации распечатывается на терминале.

Команда OFFLINE приводит к изменению состояния неактивного устройства на OFF LINE и тем самым исключает его из набора доступных ресурсов в текущей конфигурации системы. Это предотвращает дальнейший доступ к данному ресурсу. Для того чтобы перевести устройство в состояние OFF LINE, у него должен отсутствовать контекст, а программа HRC должна иметь возможность обнаружить отсутствие операций ввода-вывода за 1000 или меньше попыток.

Перевод в состояние OFF LINE конкретного контроллера устройства выполняется только после перевода в это состояние всех устройств, подсоединенных к данному контроллеру. Если связанное устройство не находится в состоянии OFF LINE, программа HRC отвергает команду OFF LINE так как такой переход может повлиять на выполнение текущих операций ввода-вывода.

Для освобождения устройства от контекста следует использовать соответствующие команды MCR.

Чтобы отсоединить устройство, подсоединенное задачей, необходимо аварийно завершить задачу командой MCR ABO.

Если устройство находится в состоянии «устройство отмечено для перехода в ON LINE» (например, в случае когда его контроллер находится в состоянии OFF LINE), команда OFF LINE переводит устройство в OFF LINE. Соответственно когда контроллер будет переведен в состояние ON LINE, система такое устройство активизировать не будет.

Если устройство находится в состоянии «устройство отмечено для перехода в OFF LINE», команда OFF LINE переводит его в состояние OFF LINE.

В команде OFF LINE может использоваться параметр ALL, позволяющий переводить в состояние OFFLINE все устройства в данной конфигурации, за исключением системного диска и терминала, выдавшего команду. Команда OFFLINE ALL необходима также перед выполнением сохранения системы. Для успешного выполнения команды OFFLINE ALL необходимо демонтировать все системные диски и ленты (кроме системного диска) и завершить работу всех терминалов, кроме пользовательского. Ф о р м а т :

```
CON> OFF [LINE] имя_устр-ва1   [,имя_устр-ва2, ..., имя_устр-ваN]
CON> OFF [LINE] ALL
```

где имя_устройства — имя устройства, переводимого в состояние OFF LINE;
ALL — параметр для перевода в состояние OFF LINE всех устройств.

Условия выполнения команды OFFLINE:

•конфигурацию системы рекомендуется изменять только с привилегированного терминала;

•устройства не должны быть присоединены к задачам;

•устройства не должны быть монтированы;

•если устройство является терминалом, он должен быть незарегистрированным.

П р и м е р :

```
CON> OFFLINE LPA, LP0
CON> OFF VZA, VZB, VZC, VZD
```

Команда ONLINE изменяет состояние устройства на ON LINE и разрешает к нему доступ.

Переход в состояние ONLINE может быть выполнен только в том случае, если имеется связь между процессором и устройством и если драйвер устройства может успешно его инициировать.

Если связь не установлена, команда ONLINE переводит устройство в состояние «устройство отмечено для перехода в ON LINE». После установки связи устройство переводится в состояние ON LINE.

В команде ONLINE может быть также использован параметр ALL, позволяющий перевести в состояние ON LINE все известные системе устройства. Ф о р м а т :

```
CON>ONL [INE]   имя_устр-ва1   [,имя_устр-ва2,..., имя_устр-ваN]
CON>ONL [INE] ALL
```

где имя_устройства — имя устройства, переводимого в состояние ON LINE;
ALL — параметр для перевода в состояние ON LINE всех устройств.

Условия выполнения команды:

•конфигурацию системы допускается изменять только с привилегированного терминала;

•устройство должно присутствовать в конфигурации системы и быть сгенерированным во время генерации;

•если устройство поддерживается загружаемым драйвером, он должен быть загружен.

Команда ONLINE активизирует устройство только в том случае, когда соответствующий контроллер находится в состоянии ON LINE. В первую очередь проверяется физическое наличие устройства на контроллере. Далее определяется геометрия устройства и его тип. Если устройство присутствует, оно становится доступным для использования.

Если контроллер находится в состоянии OFF LINE, устройству присваивается состояние «устройство отмечено для перехода в ON LINE». После установки связи определяется геометрия и тип устройства и ему присваивается состояние ON LINE.

При активизации контроллера устройства команда ONLINE проверяет регистр контроля и состояния (РКС) контроллера с целью определения его наличия. Если контроллер физически присутствует, все устройства данного контроллера, имеющие состояние «устройство отмечено для перехода в ON LINE», проверяются на физическое присутствие, определяются их геометрия и тип и переводятся в состояние ON LINE. Для отсутствующих на контроллере устройств сохраняется состояние OFF LINE.

П р и м е р :

```
CON>ONL ALL
```

Данная команда переводит в состояние ON LINE все устройства, известные системе. Если не установлена связь с устройством, оно переводится в состояние «устройство отмечено для перехода в ON LINE». Выполнение данной команды может занять несколько секунд в зависимости от конфигурации. Если устройство по каким-то причинам не может быть переведено в состояние ON LINE, сообщение не выдается.

Команда SET позволяет изменить значения вектора прерываний и адреса регистра контроля и состояния контроллера, находящегося в состоянии OFF LINE.

В команде SET не выполняется проверка задаваемых данных. Если вектор прерывания и адрес контроллера заданы неправильно, то контроллер использовать невозможно. При этом если задается вектор прерывания, используемый другим устройством, во время перевода устройства в состояние ON LINE происходит ошибка. Ф о р м а т :

```
CON > SET контроллер параметр = значение
```

где контроллер — имя контроллера устройства;
параметр — название задаваемого параметра. Параметрами могут быть:
CSR — изменение адреса регистра контроля состояния контроллера;
VEC — изменение вектора прерывания;
значение — значение вектора или адреса контроллера. Если задается параметр CSR, значение должно быть в диапазоне от 160000 до 177777(8).

Отметим, что если задать CSR= 160000, контроллер считается неизвестным и не может быть переведен в состояние ON LINE. Это позволяет применять команду CON ONLINE ALL в случаях, когда

реальная конфигурация комплекса не соответствует сгенерированной. Если задается адрес РКС, совпадающий с физически присутствующим адресом контроллера другого типа, возможен крах системы. Значение вектора должно быть меньше 774(8). Для изменения вектора и адреса устройства его состояние должно быть OFF LINE, а драйвер должен быть загружен.

П р и м е р :

```
CON > SET LPA VEC = 160
```

Данная команда изменяет значение вектора прерываний для первого контроллера устройства печати.

Обслуживание отказов. Отказы, как правило, вызывают прекращение правильного выполнения программ пользователя. К отказам относятся следующие виды ошибок:

•ошибка пользователя;

•ошибка системного программного обеспечения;

•ошибка прикладного программного обеспечения;

•отказ аппаратуры.

Ошибки пользователя (например, случайная остановка дискового накопителя), как правило, очевидны, так как они незамедлительно отражаются на поведении системы. Ошибки системного и прикладного программного обеспечения обнаруживаются чаще всего во время отладки прикладных программ.

Отказы аппаратуры можно объединить в следующие группы:

•дефекты носителей информации;

•сбой;

•временные отказы;

•корректируемые ошибки;

•некорректируемые ошибки.

Д е ф е к т ы н о с и т е л е й и н ф о р м а ц и и обнаруживаются во время выполнения запросов ввода-вывода при наличии дефектных участков на магнитных поверхностях дисков или лент. При этом драйверы автоматически повторяют выполнение операций чтения. До инициализации дисков их необходимо проверить с помощью программы BAD, которая находит дефектные блоки и предотвращает их использование. Если во время работы на носителе появляются новые дефектные блоки, их можно обнаружить, используя услуги подсистемы ERRLOG. При этом может потребоваться замена диска, если число ошибок велико.

Для устройств, поддерживаемых драйвером DUDRV, дефектные блоки могут быть заменены с помощью программы RCT.

Сбой, как правило, нельзя повторить, и он обусловливается внешними причинами, такими, как разряды статического электричества, накапливаемого на одежде или покрытии полов. Если результат сбоя повлиял на выполнение операции ввода-вывода, она автоматически повторяется, и программа пользователя этого не заметит. Подсистема ERRLOG зарегистрирует такой сбой.

Если сбой происходит в памяти или в центральном процессоре, происходит крах системы, и она должна быть перезагружена.

В р е м е н н ы е о т к а з ы обусловливаются нестандартными условиями в системе, обычно они возникают в период высокой нагрузки системы или устройства. Чаще всего это происходит из-за кратковременного отказа питания внешнего устройства.

Примером временного отказа является отказ питания накопителя на магнитных дисках, что приводит к сбросу бита достоверности накопителя. Демонтирование и повторное монтирование устройства вновь устанавливают этот бит, что позволяет продолжить выполнение операций ввода-вывода. Если источник временной ошибки не может быть обнаружен или повторение ситуации затруднено, можно воспользоваться программой IOX. Она моделирует условия высокой нагрузки, в результате чего можно определить причину временного отказа.

Временные отказы регистрируются в подсистеме ERRLOG.

К о р р е к т и р у е м ы е о ш и б к и не приводят к ошибочному завершению операции ввода-вывода. Драйвер устройства пытается выполнить операцию повторно и, в зависимости от возможностей устройства, выполняет корректировку данных с помощью кодов коррекции или изменения позиционирования головок.

Появление корректируемых ошибок свидетельствует о низком качестве покрытия или о неправильной настройке головок. В случае неверной настройки головок корректировка может быть

выполнена за счет изменения позиционирования, задаваемого через контроллер устройства.

Корректируемые ошибки регистрируются подсистемой ERRLOG.

Некорректируемые ошибки препятствуют нормальному использованию устройства. Такие ошибки обнаруживаются достаточно легко. Информация, накапливаемая в подсистеме ERRLOG, как правило, позволяет однозначно локализовать ошибку.

Корректируемые ошибки и ряд ошибок из-за дефектов поверхности носителей являются восстанавливаемыми. Они регистрируются в подсистеме ERRLOG, но не влияют на ее работоспособность.

Некорректируемые ошибки и некоторые ошибки, возникающие из-за дефектов поверхности носителей, являются невосстанавливаемыми. Они, как правило, приводят к отказу прикладного программного обеспечения. Операционная система более устойчива, так как для ее функционирования достаточно работоспособного системного диска, центрального процессора и памяти.

После отладки программного обеспечения доступность системы определяется надежностью отдельных устройств и временем, необходимым для их восстановления. ОСРВМ в некоторых случаях может обходить отказы оборудования. Для этого требуется изолировать отказавшее устройство, исправить причину ошибки и восстановить к нему доступ. В зависимости от глубины ошибки и ее местонахождения иногда требуется перезапустить прикладную программу или операционную систему.

Отказы часто проявляются таким образом, что их достаточно легко обнаружить, вследствие чего можно запретить доступ к отказавшим устройствам, чтобы не допустить краха системы. Необходимо постоянно вести наблюдения за состоянием всех периферийных устройств. Наблюдение выполняется путем периодического анализа распечаток, выдаваемых подсистемой ERRLOG, и запуска программы IOX для редко используемых устройств.

Как подсистема ERRLOG, так и программа IOX распечатывают информацию об ошибках устройств, которая может служить основанием для принятия решения об их исключении из системы.

Процедура восстановления системы зависит от режима ее использования. Если система применяется для разработки программ, следует использовать программу IOX и подсистему ERRLOG. Они помогают уточнить причину отказов и предотвратить дальнейшее их влияние на работу системы.

Если система применяется для управления в реальном масштабе времени, можно рекомендовать системный отладчик (XDT), который дает возможность локализовать и исправить ошибку. При этом главной задачей является скорейшее восстановление работоспособности с проведением минимального тестирования для принятия дальнейших решений.

При рестарте системы и прикладного программного обеспечения в первую очередь следует загрузить ОСРВМ с минимальной конфигурацией. Далее постепенно включаются в систему устройства, необходимые для работы прикладных программ. Если после включения в конфигурацию устройства обнаруживаются ошибки, его следует исключить. В результате постепенно создается конфигурация, которая позволяет запустить прикладное программное обеспечение. Таким образом, применяются следующие приемы восстановления системы после отказов:

•сбой может привести к краху системы, что требует перезагрузки;

•некорректируемые ошибки не позволяют в дальнейшем использовать устройство и требуют его исключения из конфигурации. Далее его следует проверить диагностическими тестами;

•источник временных ошибок не всегда очевиден, поэтому следует применять методику постепенного расширения конфигурации до определения устройства, вызывающего ошибки.

6.3. СРЕДСТВА УПРАВЛЕНИЯ ДИНАМИЧЕСКОЙ ПАМЯТЬЮ

Средства управления динамической памятью (пулом) системы позволяют ограничивать использование динамической памяти и сообщать оператору о ее состоянии. Эти средства состоят из двух компонентов: подпрограммы управления пулом управляющей программы и привилегированной программы управления динамической памятью PMT.

Во время генерации ОСРВМ предоставляется возможность включить в систему средства управления пулом. По умолчанию средства управления пулом в ОСРВМ включаются автоматически.

Подпрограмма управления пулом в управляющей программе контролирует количество свободной динамической памяти и оповещает об ее экстремальных состояниях (событиях) программу PMT, которая выполняет необходимые действия для восстановления нормального состояния

пула. Экстремальные состояния пула классифицируются как события «низкого» и «высокого» состояния. Подпрограмма управления пулом сигнализирует о состоянии пула установкой флага события и активизирует программу PMT. Дальнейшая реакция на состояние динамической памяти остается за программой PMT.

«Высокое» состояние — состояние, когда количество свободной памяти достаточно, чтобы поддерживать требования рабочей нагрузки системы. «Низкое» состояние — состояние, когда количество свободной памяти близко к точке, в которой работоспособность системы существенно снижается.

Событие «высокого» состояния пула объявляется при переходе от состояния нехватки динамической памяти к состоянию, когда этой памяти достаточно.

Это событие происходит, когда в системную динамическую память возвращается часть памяти, превышающая ее верхний предел. Верхний предел динамической памяти устанавливается по команде программ MCR или VMR SET/PLCTL.

Имеются следующие причины возникновения события «низкого» состояния пула:

•переход пула из «высокого» состояния в «низкое»;

•отказ выделения динамической памяти.

Истощение пула происходит, когда размер свободной памяти в пуле становится меньше нижнего предела. Отказ выделения динамической памяти происходит, когда наибольший фрагмент пула слишком мал для удовлетворения запроса. Не следует смешивать отказ выделения динамической памяти с общей фрагментацией и экстремальной фрагментацией пула.

Нижний предел динамической памяти устанавливается по команде программ MCR и VMR SET/PLCTL.

Интерфейс управления пулом — механизм, используемый управляющей программой для определения экстремальных событий в пуле. Подпрограмма, реализующая данный интерфейс, включена в модуль CORAL.MAC в каталоге [11. 10] дистрибутива OCPBM. Интерфейс функционирует следующим образом:

•предоставляет управляющей программе возможность передать информацию программе PMT;

•предоставляет управляющей программе возможность получить реакцию или подтверждение от программы PMT. (Подтверждение может быть использовано управляющей программой для управления дальнейшей передачей информации к программе PMT.)

События пула обусловливают переход либо в «высокое», либо в «низкое» состояние. В свою очередь «высокое» и «низкое» состояния в сочетании со специфическим уровнем фрагментации создают «высокое» или «низкое» состояние пула.

Программа PMT транслируется, строится и устанавливается во время генерации системы. После загрузки системы программа PMT активизируется управляющей программой. Если функции стандартной программы PMT, поставляемой в составе OCPBM, не соответствуют требованиям пользователя, можно написать и включить в систему собственную программу PMT.

Первая функция программы PMT — реагировать на состояние пула. Состояние, на которое реагирует программа PMT, обнаруживается и передается подпрограммой управления пулом управляющей программой.

Вторая функция программы PMT — управление фрагментацией пула. Программа PMT реагирует на состояние и экстремальную фрагментацию пула. Реакция PMT изменяется в зависимости от текущего уровня фрагментации. На реакцию программы PMT влияют два уровня фрагментации:

•если наибольший фрагмент пула равен или больше, чем минимально требуемый, то программа PMT реагирует только на изменение состояния пула, обнаруживаемого управляющей программой;

•если наибольший фрагмент становится меньше минимально требуемого, программа PMT рассматривает это как событие «низкого» состояния пула. Если система еще не находится в «низком» состоянии пула, названное событие вызывает переход в «низкое» состояние, и программа PMT реагирует соответственно этому состоянию.

При «низком» состоянии пула управляющая программа передает об этом сообщение программе PMT. Программа PMT определяет, является ли изменение пула, отмеченное управляющей программой, временным. Изменение является временным, если в момент реакции программы PMT на сигнал управляющей программы стало достаточно свободной памяти, чтобы превысить значение верхнего предела. Если изменения временные, PMT игнорирует сигнал и продолжает исполнять действия в

соответствии с «высоким» состоянием пула. Если изменения не временные, PMT заключает, что состояние пула «низкое», и выполняет действия, соответствующие «низкому» состоянию пула.

При «высоком» состоянии пула управляющая программа передает об этом сообщение программе PMT. Программа PMT определяет, является ли изменение, сообщенное управляющей программой, временным. Изменение считается временным, если в момент реакции программы PMT общее количество свободной динамической памяти меньше значения верхнего предела. Если изменение временное, программа PMT игнорирует сигнал и продолжает исполнять действия по «низкому» состоянию. Если изменение не временное, программа PMT инициирует действия, соответствующие «высокому» состоянию пула.

Экстремальной является такая фрагментация, при которой наибольший кусок пула равен не более 84 байтам (десятичное значение). В этом случае динамическая память почти исчерпана и ее невозможно освободить с помощью незапланированного ввода команды с терминала, например команды программы MCR ABORT. Когда происходит экстремальная фрагментация пула, программа PMT выводит соответствующее сообщение и распечатывает список задач на аварийное завершение на консольный терминал.

```
DD—MMM—YY HH:MM:SS— —WARNING — — — FREE POOL EXHAUSTED
                              (предупреждение — — пул исчерпан)
ABORTABLE TASKS IN MEMORY:
(удаляемые задачи в памяти:)
TTTTTT P I/O = XXX. TTNN:
```

где TTTTTT — имя аварийно завершаемой задачи;
P — указатель, что задача привилегированная, если пробел — непривилегированная;
XXX — счетчик ввода-вывода задачи;
TTNN: — терминал, с которого запущена задача.

Программа PMT распечатывает задачи в порядке уменьшения их приоритета, но не выводит приоритет отдельных задач. Затем программа PMT подсказывает, что следует выбрать одну задачу из списка:

```
ENTER A TASK TO ABORT, OR PRESS RETURN TO EXIT:
(выберите удаляемую задачу или нажмите  <CR>  для  выхода)
```

Чтобы изменить реакцию программы PMT, необходимо отредактировать файл построения PMTBLD.CMD и выполнить следующие действия:

- удалить задачу PMT (имя задачи PMT...), используя VMR;
- отредактировать файл построения;
- перестроить программу PMT;
- установить новый образ задачи PMT, используя VMR.

Когда система будет повторно загружена, начнет выполняться новая задача PMT. Если, однако, необходимо изменить образ задачи PMT, следует сначала аварийно завершить задачу PMT, удалить и перестроить программу PMT, используя команду программы MCR.

Можно изменить следующие параметры программы PMT:

- временные интервалы;
- реакцию (действия);
- управление использованием пулом.

Чтобы аварийно завершить задачу PMT, необходимо выполнить одно из двух действий:

- выполнить команду ABORT с привилегированного терминала;
- выполнить директиву аварийного завершения из привилегированной задачи.

6.4. ПРОГРАММА УПРАВЛЕНИЯ ЗАМЕНОЙ ДЕФЕКТНЫХ БЛОКОВ (RCT)

Программа RCT выполняет замену и восстановление дефектных блоков на дисках, поддерживаемых протоколом управления массовой памятью (ПУМП). Обработка дефектных блоков согласно протоколу ПУМП производится в четыре этапа:

- обнаружение дефектного блока;
- регистрация дефектного блока;
- замена дефектного блока;
- повтор выполнения операции.

Контроллер дисков, поддерживающий ПУМП, обнаруживает дефектный блок и сообщает об этом драйверу DUDRV. Драйвер, получив это сообщение, активизирует программу RCT, которая выполняет все функции по замене дефектного блока. Это позволяет контроллеру повторно выполнить операцию ввода-вывода на замененном блоке.

Программа RCT также выполняет замену и восстановление блоков на дисковых накопителях, поддерживаемых протоколом ПУМП, которые перешли в автономный режим во время операции замены или во время записи содержимого блоков, находящихся в дисковом кэше. Программа RCT устанавливается в ОСРВМ с именем задачи RCT...

После перевода диска в комплексный режим программа RCT осуществляет его проверку на завершение замены дефектных блоков и на разрушение кэша записываемых блоков. Если замена дефектных блоков во время перехода диска в автономный режим выполнена не полностью, программа RCT завершает этот процесс замены. Если во время перехода диска в состояние OFF LINE кэш диска переписан из памяти не полностью, программа RCT устанавливает для данного тома режим «только чтение».

Если во время генерации системы было выбрано устройство DU:, то построение драйвера и задачи RCT..., а также их установка в системе осуществляются автоматически. Необходимость выполнить эти операции вручную может возникнуть в двух случаях:

•если оператор удалил задачу RCT... или выгрузил драйвер DUDRV;

•из командного файла SYSVMR.CMD удалены команды установки программы RCT или загрузки драйвера.

При ручной установке задачи RCT... и загрузке драйвера DUDRV необходимо в первую очередь установить задачу RCT... В противном случае на консольный терминал будет выдано сообщение, что задача RCT... не установлена.

Для установки задачи RCT... можно воспользоваться командой

```
>INS $RCT/TASK = RCT...
```

Пространство диска, поддерживаемого ПУМП, разбито на три части:

•пространство логических блоков (LBN);

•пространство блоков замены (RBN);

•таблицы управления переназначением.

Пространство LBN доступно задаче пользователя для обмена информацией между диском и операционной системой. Когда контроллер диска обнаруживает дефектный логический блок, он сообщает об этом драйверу, который активизирует задачу RCT... для замены логического блока.

Блоки замены (RBN) — это блоки на диске, специально резервированные для замены дефектных логических блоков. Во время нормальной работы с диском пользовательские задачи не имеют доступа к блокам замены.

Таблицы управления переназначением доступны только задаче RCT... (через драйвер DUDRV) и контроллеру устройства. Каждый элемент таблицы содержит ссылку к одному блоку замены и описание его состояния:

ALLOCATED (занято) — данный RBN в настоящее время занят;
UNALLOCATED (свободно) — данный RBN не используется;
UNUSABLE (недоступен) — RBN не может быть использован;
PRIMARY (первичный) — первый RBN доступен для чтения при замене дефектного блока;
SECONDARY (вторичный) — следующий RBN доступен для чтения.

Для данного LBN первичный RBN всегда находится в некотором предопределенном месте на магнитном диске, например в последнем секторе той же дорожки, где размещен LBN.

Для данного LBN все другие RBN являются вторичными.

Нулевой сектор таблиц управления переназначением содержит слово флагов, позволяющее определить необходимость проведения процедуры восстановления во время перевода диска в состояние OFF LINE.

Обслуживающие программы BAD, BRU и INI воспринимают устройства, поддерживаемые ПУМП, как устройства без последней дорожки. При этом программа RCT не заменяет программу BAD. Программа BAD может быть использована для создания файла дефектных блоков, который не будет содержать информации о дефектных блоках, обнаруженных контроллером устройства и замененных программой RCT.

6.5. СИСТЕМА РЕЗЕРВИРОВАНИЯ ДИСКОВ (СРД)

СРД обеспечивает одновременное выполнение записи на два диска с файловой структурой ОСРВМ.

СРД создает идентичные данные на двух дисках, называемых резервированной парой. В системе может быть создано более одной резервированной пары, но такие пары не могут перекрывать друг друга. Первый диск пары, так называемый *первичный диск,* является исходным диском, который существует независимо от активности СРД. Второй диск пары (*вторичный диск*) подготавливается с помощью специальных команд СРД и представляет собой точную копию первичного.

Команда START активизирует задачу управления резервирования с именем SHADDN (или SHDDNN), где DDNN является именем первичного устройства. Данная задача выполняет копирование первичного диска на вторичный. Запись на оба диска осуществляется параллельно с процессом копирования. После ввода команды START, СРД обеспечивает динамическую запись данных на вторичный диск по мере их записи на первичный.

Запись данных одновременно на два диска применяется в следующих случаях:

•для дублирования важной информации с целью предотвращения непреднамеренного разрушения. С помощью СРД критические данные дублируются на двух дисках. Эта избыточность позволяет сохранить данные в случае возникновения ошибки на диске и тем самым уменьшить время восстановления системы. В результате сокращается время неработоспособности системы, так как ошибки на диске не обязательно будут приводить к остановке прикладных программ;

•в системе постоянно имеется свежая копия первичного диска, что предотвращает трату времени и ресурсов для его защиты.

СРД работает без вмешательства пользователя, поскольку все операции по дублированию выполняются управляющей программой ОСРВМ. Управляющая программа производит одновременную запись как на первичный, так и на вторичный диски. При выполнении операций чтения в первую очередь осуществляется попытка чтения первичного диска. Если происходит ошибка чтения, управляющая программа выполняет операцию чтения со вторичного диска. Все ошибки, возникающие во время работы с парой дисков, регистрируются на консольном устройстве.

Для запуска СРД необходимо иметь по крайней мере два дисковых пакета одного типа, которые должны содержать идентичную информацию о дефектных блоках.

Для подготовки к работе СРД необходимо для каждой пары выполнить следующие действия (предполагается, что СРД включена в ОСРВМ во время генерации системы):

1) образовать пары устройств одного типа, например два СМ5408. Один из этих дисков будет содержать первичный диск, который будет резервироваться другим (вторичным). Например, если в системе имеется 4 накопителя СМ5408, а системным является DM0: (первичный), в качестве вторичного диска может быть выбрано одно из трех устройств: DM1:, DM2:, DM3:;

2) образовать набор (три или более) дисковых пакетов, которые будут использоваться для резервирования. При создании набора следует руководствоваться двумя правилами:

•диски, используемые для СРД, должны быть логически эквивалентными, т. е. иметь идентичные файлы дефектных блоков. Для этого следует выбрать один диск в качестве первичного и образовать для него файл дефектных блоков, являющийся логической суммой всех дефектных блоков дисков набора. После этого, во время запуска СРД, содержимое первичного диска, включая информацию о дефектных блоках, копируется на вторичный. В дальнейшем управляющая программа записывает те же данные как на первичный, так и на вторичный диски. При этом операции записи на вторичный диск будут производиться только в работоспособные блоки, так как информация в файле дефектных блоков на первичном диске включает и дефектные блоки вторичного диска;

•в качестве вторичного может быть использован и другой диск из набора логически эквивалентных дисков. Такой подход позволяет уменьшить время, необходимое для восстановления системы;

3) для подготовки набора логически идентичных дисков необходимо запустить программу BAD с ключом /LI для всех дисков набора и получить распечатку всех дефектных блоков. Данная распечатка содержит всю информацию, необходимую для создания общего файла дефектных блоков на первичном устройстве.

Файлы дефектных блоков на первичном устройстве создаются с помощью ключа /BAD=[MAN] в команде оператора INITVOLUME следующим образом:

```
>INI DDNN:/BAD = [MAN]
```

После нажатия возврата каретки INI выдает следующую подсказку:

INI >LBN(S) =

В ответ на подсказку необходимо ввести с терминала список дефектных блоков, обнаруженных программой BAD на всех дисках набора. Созданный файл дефектных блоков в дальнейшем будет использован программами DSC и BRU на данном томе;

4) выполнить копирование данных исходного диска на первичный. При этом исходный диск нельзя использовать в качестве диска для резервирования, так как его файл дефектных блоков не содержит информации о других дисках набора;

5) один из дисков набора выбирается в качестве вторичного и монтируется как «чужой том» с помощью команды:

>MOUNT DDNN:/FOR

Для управления СРД предоставляется пять команд, которые позволяют запустить, остановить, аварийно закончить или продолжить работу СРД. Кроме того, имеется команда, позволяющая отобразить состояние СРД. Команды могут быть введены только с привилегированного терминала двумя способами:

в виде аргументов команды

>SHADOW START

в ответ на запрос программы

>SHA
SHA>START

Команда START выполняет следующие действия:

•проверяет, что первичный диск является томом с файловой структурой OCPBM;

•проверяет, что вторичный диск монтирован в системе с ключом /FOR;

•проверяет, что первичный и вторичный диски одного типа;

•создает структуру данных для СРД (UMB), которая используется управляющей программой при операциях параллельной записи;

•начинает дублирование данных из первичного диска на вторичный.

Команда START имеет следующий формат:

START DDNN:TO DDXX:

где DDNN: — имя первичного диска с файловой структурой OCPBM;
DDXX: — имя вторичного диска, который используется в качестве резервного.

Команда START копирует содержимое первичного диска на вторичный. После окончания работы команды START вторичный диск становится логической копией первичного, так как файл дефектных блоков на обоих дисках идентичен. Это означает, что все блоки данных, дефектные блоки, заголовки, каталоги и другие структуры занимают одни и те же места на первичном и вторичном дисках.

Оба диска имеют эквивалентные логические номера блоков, а в случае системного диска оба являются загружаемыми с помощью аппаратного загрузчика. Кроме того, после выдачи команды START вторичный диск становится диском с файловой структурой OCPBM, так как команда выполняет точное копирование первичного диска на вторичный. Если во время копирования на вторичный диск (после выдачи команды START) на первичный диск выполняется запись новых данных, они автоматически записываются и на вторичный диск. При этом чтение со вторичного диска может быть осуществлено только для тех блоков, которые уже переписаны с первичного. Информация о последнем скопированном блоке содержится в блоке UMB. Таким образом, резервная копия на вторичном диске считается готовой только после завершения работы команды START. Дальнейшее вмешательство пользователя необходимо лишь в случае останова СРД.

Команда STOP выполняет следующие действия:

•проверяет, что первичное устройство резервируется вторичным;

•если нет запросов ввода-вывода, освобождает структуру данных UMB и останавливает резервирование для данной пары;

•отмечает на удаление структуру UMB, если запросы ввода-вывода не завершены.

Ф о р м а т команды:

STOP DDNN: где DDNN: — имя первичного диска с файловой структурой OCPBM.

Команда STOP блокирует дальнейшую работу СРД, при этом, если не завершилась работа команды START, команда STOP будет отвергнута с выдачей соответствующего сообщения. Для того чтобы остановить СРД во время работы команды START, можно воспользоваться командой ABORT.

Отсутствие незавершенных операций ввода-вывода не всегда означает, что все действия с файловой системой закончены. Оператор должен убедиться, что все задачи, которые записывают данные на резервированную пару, завершились и все файлы, открытые для записи, закрыты. Только после этого можно выдавать команды STOP и ABORT.

Команда ABORT выполняет следующие действия:

•проверяет, резервируется ли первичное устройство вторичным;

•прекращает процесс копирования и выполняет команду STOP;

•останавливает параллельную запись даже в случае активного процесса копирования.

Ф о р м а т команды:

ABORT DDNN: где DDNN — имя первичного диска.

Команда CONTINUE выполняет следующие действия: •проверяет, имеет ли первичный диск файловую структуру ОСРВМ и смонтирован ли он в системе;

•проверяет, смонтирован ли вторичный диск с ключом /FOR;

•проверяет, имеют ли оба диска один и тот же тип;

•создает структуру данных UMB и начинает работу СРД.

Ф о р м а т команды:

CONTINUE DDNN:TO DDXX:

где DDNN: — имя первичного диска;
DDXX: — имя вторичного диска.

Команда CONTINUE продолжает работу СРД после ее остановки с помощью команды STOP. Команда CONTINUE предполагает, что оба диска являются логически идентичными и не проверяет их на эквивалентность. Оператор должен быть уверенным в том, что после выдачи команды STOP ни на первичный, ни на вторичный диски запись не производилась.

Команда DISPLAY позволяет распечатать имена первичных и вторичных устройств, на которых в данный момент времени активна СРД. Ф о р м а т :

DISPLAY

Резервируемые пары устройств распечатываются в следующем виде:

UMB	PRIMARY	SECONDARY
XXXXXX	DDNN:	DDXX:

где XXXXXX — адрес структуры данных UMB для данной пары. Данная структура данных резервируется в области динамического пула в адресном пространстве управляющей программы;
DDNN: — имя первичного диска;
DDXX: — имя вторичного диска.

Во время работы СРД управляющая программа осуществляет запись данных из буфера задачи сначала на первичный, а затем на вторичный диски. Если во время записи на одном из дисков происходит ошибка, информация об этом будет получена только во время попытки чтения ошибочного блока. Для того чтобы получить сообщение об ошибке сразу после записи, следует одновременно с работой СРД задать проверку записи на обоих дисках с помощью следующих команд:

>SET /WCHK = DDNN:
>SET /WCHK = DDXX:

При этом необходимо учитывать, что операция записи с проверкой выполняется дольше, чем просто запись.

Если задана операция «запись с проверкой», СРД информирует пользователя об ошибках по мере их появления выдачей сообщений на консольном терминале. Сообщения об ошибках содержат номера устройств и логических блоков.

Во время операции чтения управляющая программа в первую очередь обращается к первичному диску. Если во время выполнения операции на первичном диске происходит ошибка, делается попытка прочитать тот же блок со вторичного диска. Если и на вторичном диске операция чтения выполнена

неуспешно, выдается сообщение об ошибке на вторичном диске. Это одновременно обозначает, что и на первичном диске была ошибка чтения.

Ошибка в одном и том же месте на обоих дисках весьма маловероятна и восстановление данных после такой ошибки невозможно. Если ошибки происходят на обоих дисках в том же или в разных местах, единственной возможностью восстановления данных является остановка СРД и пользовательских задач. Далее необходимо подобрать новый первичный и вторичный диски, создать файл дефектных блоков на первичном диске и переписать данные из старого первичного диска на новый.

Если ошибка произошла на разных блоках дисковой пары, информация может быть сохранена с помощью программы BRU. При этом в качестве вводного устройства необходимо использовать пару резервированных дисков, тогда СРД обеспечит чтение данных с того диска, с которого они могут быть прочитаны без ошибок.

Программа BPU не позволяет создать новый первичный диск. Для этого необходимо применить программу BAD.

6.6. ПОДСИСТЕМА УЧЕТА ИСПОЛЬЗОВАНИЯ РЕСУРСОВ

Подсистема учета использования ресурсов (ПУИР) ОСРВМ — средство создания файла транзакций, содержащего статистическую информацию о работе системы и ее составных частей. Данная подсистема включается в ОСРВМ во время генерации.

Информация, накапливаемая в файле транзакций, может использоваться для учета, распределения и оплаты машинного времени, а также для анализа эффективности работы системы в целом.

ПУИР обеспечивает сбор данных о работе как отдельных пользователей, так и всей системы. Файл транзакций содержит, в частности, следующие данные: данные пользователя:

идентификатор сессии, терминал и номер счета;
дата и время окончания оплаты за время;
время использования процессора;
список активных задач во время выхода из системы или ее краха;
число запусков задач;
число выданных директив и запросов QIO$;
состояние при выходе из системы или при ее крахе; данные задачи;
идентификатор сессии, терминала и номера счета;
число загрузок перекрытий;
число выгрузок задачи;
наибольший приоритет задачи;
время начала и конца работы задачи;
время использования процессора;
число выданных директив и запросов QIO$;
состояние при выходе из системы или при ее крахе; данные системы;
время запуска подсистемы учета;
время остановки подсистемы учета (0, если система остановлена аварийно);
код закрытия системы;
ID файла транзакции, последовательный номер, устройство;
время и дата последнего сканирования и период сканирования в секундах;
общее время центрального процессора;
общее число выполненных задач;
общее число входов пользователей в систему;
текущее число пользователей;
число выгрузки задач;
число запусков задачи SHF;
число выданных директив;
число запросов QIO$;

данные о регистрации пользователей:

имя пользователя, идентификатор сессии, терминал и номер счета;
код идентификации пользователя UIC, дата и время;

ошибочный пароль;
данные о подсоединении, отсоединении и демонтировании устройств:
идентификатор сессии, терминал, номер счета;
устройство, дата и время подсоединения или отсоединения устройства;
данные о монтировании устройств:
идентификатор сессии, терминал и номер счета;
устройство, дата и время его монтирования;
метка тома для устройств с файловой структурой ОСРВМ;
тип монтирования (чужой, разделяемый и т. д.);
UIC пользователя;
код защиты тома;
имя файлового процессора для устройства;
данные задачи системной печати:
идентификатор сессии, терминал и номер счета;
дата и время печати;
имя задания и число страниц;
число выводимых файлов, устройство печати и число копий;
приоритет задания;
данные об изменении времени:
старое время;
новое время;
данные об использовании устройств:
счетчики операций ввода-вывода;
счетчики ошибок;
служебная информация.

Вывод информации из файла транзакций осуществляется тремя способами:

1) с помощью команды SHOW ACCOUNTING:

SHOW ACCOUNTING/INFORMATION (распечатывает информацию о терминалах);
SHOW ACCOUNTING/TRANSACTION_FILE (распечатывает сформатированную информацию из файла транзакций);

2) с помощью программы, написанной с применением системы управления данными;
3) с помощью программы, написанной пользователем.

ПУИР собирает информацию о работе систему в блоках вторичного пула, а затем записывает каждый блок в виде записи в файл транзакций. Используются следующие типы блоков:

•блок учета задач (TAB);
•блок учета пользователей (UAB);
•блок учета системы (SAB);
•блоки учета транзакций.

При описании команд ПУИР применяются такие соглашения:
параметры разделяются пробелами;
сокращение имен параметров допускается;
ПУИР запрашивает пропущенные параметры.
Все команды, кроме SHOW ACCOUNTING/INFORMATION, являются привилегированными.
ПУИР поддерживает следующие команды:
START/ACCOUNTING — запускает ПУИР;
SET ACCOUNTING — модифицирует параметры в работающей ПУИР;
STOP/ACCOUNTING — останавливает ПУИР;
SHOW ACCOUNTING/INFORMATION — распечатывает учетную информацию о работе терминалов;
SHOW ACCOUNTING/TRANSACTION_FILE — преобразует файл транзакций в символьный формат для отображения на терминале или вывода в файл;
SHOW ACCOUNTING/DATATRIEVE — преобразует файл транзакций в файл, пригодный для обработки системой управления данными ФОБРИН.

Команда START/ACCOUNTING активизирует работу ПУИР. Данная команда может быть выполнена только после установки задач SYSLOG и ...ACC. Ф о р м а т :

START/ACCOUNTING [параметр] [параметр]

Допустимыми параметрами являются:

FILE: спец.файла — имя файла транзакций, который будет создан ПУИР для хранения данных. По умолчанию используется LB: [1, 6] ACNTRN.SYS;

EXTEND_SIZE:число — размер сегмента файла транзакций при его создании и размер, на который он увеличивается при необходимости. Если активность ПУИР низкая, лучше использовать небольшой размер. При высокой активности большой размер сегмента уменьшает накладные расходы системы. По умолчанию размер равен 10 блокам;

POOL_RESERVE:число — число в данном параметре указывает, сколько свободных блоков должно быть оставлено во вторичном пуле после их резервирования ПУИР. Если свободного места после запроса ПУИР не остается, резервирование аннулируется, что предотвращает перегрузку вторичного пула. По умолчанию «неприкосновенный запас» равен одной четвертой величины вторичного пула;

SYSTEM_STATISTICS: YES/NO — данный параметр указывает на необходимость накопления информации о работе системы. Работа системы регистрируется в блоках SAB и содержит следующие поля данных:

B. CPU — общее время использования центрального процессора;

B. DIR — общее число выполненных в системе директив;

B. QIO — общее число операций QIO$;

B. TAS — общее число задач.

По умолчанию накопления информации о работе системы разрешено:

STATISTICS_SCAN [: число] — число, заданное в данном параметре, указывает период сбора статистики о работе устройства. Главным образом полученная информация используется для измерения параметров оптимизации позиционирования. Кроме того, предоставляется полная информация о производительности и загрузке дисков. Такая статистика может быть собрана либо однократно, либо периодически. Для однократного сбора число не задается. Период сбора статистики задается в формате NM (в минутах) или NS (в секундах). По умолчанию принимается 1 час. При этом должен быть задан параметр SYSTEM__STATISTICS: YES.

Для однократного сбора статистики параметр SYSTEM_STATISTICS задавать не обязательно. Периодический сбор статистики о работе устройства прекращается двумя способами:

заданием периода сбора, равным 0;

заданием параметра SYSTEM_STATISTICS:NO;

SCAN_RATE:число — число, заданное в данном параметре, указывает период записи блоков SAB и UAB во временный файл LB: [1, 6] SYSSCAN. TMP с целью его защиты в случае краха системы. Число задается в формате NM (в минутах) или NS (в секундах). Если задать нуль, периодическая запись производиться не будет, и в случае краха системы все данные будут потеряны. По умолчанию период равняется 5 мин. Если при запуске ПУИР обнаруживается временный файл, его содержимое переписывается в файл транзакций. При распечатке файла транзакций в листинг включается информация о крахе системы. Копирование данных из файла LB: [1, 6] SYSSCAN.TMP осуществляется до начала регистрации новых данных. При остановке ПУИР временный файл удаляется;

CRASH_REASON: YES/NO — если в данном параметре указано YES и ПУИР запускается после краха системы, во время которого система учета была активной, то запрашивается ввод причины краха. Длина вводимого ответа не должна превышать 60 символов. Причина краха хранится в блоке транзакции восстановления системы. Если система была остановлена нормально, ввод причины краха не запрашивается. По умолчанию используется NO;

TASK: YES/NO — если в данном параметре указано YES, то активизируется накопление статистики о работе задач. Полученные данные можно использовать при анализе периодов пиковой нагрузки системы для выявления задач, наиболее интенсивно использующих ресурсы системы. Указание NO в данном параметре прекращает накопление статистики о работе задач. По умолчанию используется NO.

Статистику о работе задач следует собирать только в случае крайней необходимости, так как учет задач требует значительного пространства на диске и файл транзакций увеличивается очень быстро.

В примере показаны данные, накапливаемые ПУИР о работе устройств о файле транзакций.

Пример:

```
DEVICE STATISTICS
TIME OF DEVICE STATISTICS = 15—DEC—88 16 27:23 DEVICE = DR1:
1) 10 COUNT=013832.
2) WORDSTRANSFERED  COUNT=9002804
3) SOPT ERROR LIMIT = 8
4) SOFT ERROR COUNT = 0
5) HARD ERROR LIMIT = 5
6) HARD ERROR COUNT = 0
7) CYLINDER CROSSED COUNT=1053
8) CURRENT FAIRNESS COUNT=0
9) FAIRNESS COUNT LIMIT=10
```

В примере поля имеют следующие значения: 1)— число запросов ввода-вывода; 2) — число переданных слов; 3) — регистрируемое число корректируемых ошибок; 4) — число корректируемых ошибок; 5) — регистрируемое число некорректируемых ошибок; 6)— число некорректируемых ошибок; 7)— количество переходов между цилиндрами; 8 — текущий счетчик приостановки в очереди оптимизации; 9) — максимальный счетчик приостановки в очереди оптимизации.

Команда SET ACCOUNTING позволяет изменять значения следующих параметров работы ПУИР после ее запуска:

• имя файла, в который записываются транзакции;
•размер сегмента;
•период записи во временный файл;
•период сбора статистики устройств;
•разрешение/запрет сбора статистики о работе задач.

Ф о р м а т команды:

SET ACCOUNTING/ [параметр] [параметр]

Возможно применение следующих параметров, синтаксис и значение которых описаны выше:

FILE [: спец.файла]
EXTEND_SI2E: число
STATISTICS_SCAN; число
SCAN_RATE: число
TASK: YES/NO

Команда STOP/ACCOUNTING останавливает ПУИР. В команде должен быть указан параметр, указывающий причину остановки, который запоминается в блоке учета работы системы SAB. Ф о р м а т ы :

STOP/ACCOUNTING причина
STOP/ACCOUNTING CLEAN_UP

В качестве параметра, указывающего на причину остановки, используются следующие ключевые слова:

MAINTENANCE — указывается при остановке системы для проведения обслуживания;
REBOOT — указывается при необходимости перезагрузки системы;
SCHEDULED_SHUTDOWN — указывается при нормальном завершении работы системы или ее использовании в режимах, не допускающих работы ПУИР;
SHUTUP — по умолчанию программа SHUTUP использует данную причину, так как других способов ее определения нет;
OTHER — задается во всех других случаях.

Параметр CLEAN_UP используется в случае, когда аварийно завершается задача SYSLOG или происходит фатальная ошибка в ПУИР. Это позволяет перевести в исходное состояние все структуры данных для перезапуска ПУИР. Подсистема воспринимает эту команду, только если это необходимо (т.е. после аварийного завершения SYSLOG или после фатальной ошибки ПУИР).

При использовании данной команды система считает все данные ПУИР некорректными, освобождает память во вторичном пуле и не модифицирует файл транзакций. В случае разрушения данных во время освобождения памяти во вторичном пуле может произойти крах системы.

Команда SHOW ACCOUNTING/INFORMATION позволяет отображать на терминале статистическую информацию из ПУИР. Непривилегированные пользователи могут получить данные только о своей работе. Привилегированному пользователю доступны любые данные. Ф о р м а т :

SHOW ACCOUNTING/INFORMATION [параметр]

Д о п у с т и м ы м и п а р а м е т р а м и являются:

TTN: — позволяет привилегированному пользователю отобразить учетную информацию о работе заданного терминала. Непривилегированный пользователь может получить информацию только о работе своего терминала. По умолчанию используется TI:;
CO: — позволяет отобразить учетную информацию о работе системных задач;
SYS — позволяет отобразить обобщенную информацию о работе системы;
TASK = имя задачи — позволяет отобразить учетную информацию о работе задачи, имя которой задано в параметре. При этом сбор статистики о работе задач должен быть разрешен.

Команда SHOW ACCOUNTING/TRANSACTION_FILE позволяет преобразовать и сформатировать данные в файле транзакций для их отображения на терминал. Результат преобразования выводится на терминал или в заданный файл. Ф о р м а т :

SHOW ACCOUNTING/TRANSACTION_FILE '[; INPUTSPEC] OUTPUTSPEC

где INPUTSPEC — спецификация файла транзакций. Если он не задан, используется спецификация по умолчанию (LB: [1, 6] ACNTRN. SYS);

OUTPUTSPEC — спецификация файла для вывода данных. Для вывода на терминал указывается TTN: или TI:

В примере приводится формат распечатки ПУИР в ответ на команду SHOW ACCOUNTING/TRANSACTION_FILE. По умолчанию в качестве файла транзакций используется файл, заданный в команде START/ACCOUNTING.

При работе ПУИР требует ряда системных ресурсов, величина которых зависит от заданного режима, числа активных терминалов, задач и т. д.

Особое внимание следует уделять режиму сбора статистики о работе задач, так как при этом требуется достаточно большое
пространство на диске и интенсивно используется вторичный пул.

Если при генерации ОСРВМ включается поддержка ПУИР, она приводит к резервированию 532 слов в адресном пространстве управляющей программы. В системном пуле резервируется пространство для заголовков одной активной задачи и одного открытого файла.

```
ACCOUNTING DATA -CURRENT FILE     11-DEC-85 16:56:04

SYSTEM CRASH RECORD
1)  TIME OF LAST SCAN=7-DEC-87 10:48:17 SCAN RATE (SEC) =300
    RESTART TIME - 7-DEC-87 19:58:16
2)  REASON =

3)  USER - ALPHA  P
          SESSION ID = F1143    TI:=TT40:        ACCOUNT = 1
          LOGIN UIC = [7, 111]  LOGGED ON = 7-DEC-87 08:58:53
          LOGGED OFF = 00-000-00 00:00:00 BILLING STOPPED=7-DEC-87 10:48:17
          CPU =4461. TASKS ACTIVE = -1       TASKSRUN=95.
          QIOS=6215.                         DIRECTIVES = 10240.
    STATUS      ACT   CRH


    USER-SYSTEM TASKS
          SESSION ID = $SY0     TI:=CO:          ACCOUNT = 0.
          LOGON UIC = [0, 0]      LOGGED ON = 6-DEC-87 15:48:16
          LOGGED OFF = 00-00-00 00:00:00 BILLING STOPPED = 7-DEC-87 10:48:17
          CPU = 103866. TASKS ACTIVE =9.     TASKS RUN =61.
          QIOS= 128967.        DIRECTIVES = 667250.
    STATUS      ACT CRH
4)  TOTAL SYSTEM STATISTICS
          ACCOUNTING STARTED =6-DEC-87 15:48:16
          ACCOUNTING STOPPED = 00-000-00 00:00:00       SHUTDOWN CODE 0.
          TRANS FILE ID = 1422            SEQ NUM = 1             DEVICE = DB1:
          TIME OF LAST SCAN = 7-DEC-87 10:48:17           SCAN RATE (SFC) =300.
          TOTAL CPU = 441264.             ZERO CPU INTERVALS - 374665.
          TOTAL TASKS = 3236. TOTAL LOGONS = 60.
          CURRENT USERS = 27.           CHECKPOINTS = 23.
          SHF RUNS=0.          DIRECTIVES = 2373564.            QIOS = 556015.
    ACCOUNTING STARTUP
          ACCOUNTING STARTED = 7-DEC-87 19:58:16
          LOGIN-BETA              B
          SESSION ID = RXP1         TI:=TT26:          ACCOUNT =0.
          LOGON UIC = [7, 102]      TIME = 7-DEC-87 19:58:35
    LOGIN -ALPHA        P
          SESSION ID = F112         TI:=TT55:          ACCOUNT = 1.
          LOGON UIC = [7, 111]      TIME = 7-DEC-85 20:05:44
    DEVICE MOUNT
          SESSION ID = F112         TI:=TT55:          ACCOUNT = 1
          TIME = 7-DEC-87 20:06:37     DEVICE = DR1:       VOL ID =WELMACK
          OWNER UIC = [0, 0]        ACP NAME = FOCACP VOL PROT MASK = 0
    TOTAL SYSTEM STATISTICS
          ACCOUNTING STARTED = 7-DEC-87 19:58:16
          ACCOUNTING STOPPED = 7-DEC-87 20:13:21    SHUTDOWN CODE 4
          TRANS FILE IN = 205 SEQ NUM = 20      DEVICE = DR0:
          STATISTICAL SCAN RATE  (SEC,) =3600,          FILE EXT, SIZE= 10.
          TIME OF LAST SCAN = 7-DEC-87 20:13:17         SCAN RATE (SEC) =,300.
          TOTAL CPU = 4260.         ZERO CPU INTERVALS = 3063.
          TOTAL TASKS = 243.        TOTAL LOGONS = 7.
          CURRENT USERS = 1.        CHECKPOINTS = 0.
          SHF RUNS = 0.                DIRECTIVES = 10241.  QIOS = 4946.
```

Для каждого пользователя, зарегистрированного в данный момент в системе, требуются два блока вторичного пула (32 слова в каждом блоке) для размещения UAB.

Для каждой работающей в системе задачи требуются два блока для хранения TAB (если разрешен сбор статистики о работе задач), для учета работы системы — два блока для SAB и два блока для системного UAB. Для регистрации операций монтирования и других однократных событий нужен один блок на короткий промежуток времени. Ф о р м а т сообщения ПУИР:

HH: MM:SS AAATNN ∗AAAAAA∗ сообщение...

где HH:MM:SS — время выдачи сообщения;
AAA — первые три буквы имени задачи;
TNN — номер терминала, выполняющего данную задачу;
NN — номер сообщения. Этот номер облегчает поиск сообщения в данном описании;
∗AAAAAA∗ — класс сообщения, определяющий его важность. Если класс отсутствует, сообщение считается информационным. Всего имеются четыре класса сообщений:

∗DIAGH∗ — информационное сообщение;
∗WARN∗ — предупреждающее сообщение;
∗ERRORH∗ — команда не выполнена;
∗FATAL∗ — фатальная ошибка, дальнейшая работа невозможна.

6.7. СИСТЕМА УПРАВЛЕНИЯ КЭШИРОВАНИЕМ ДИСКОВ

Кэширование дисков — это возможность ОСРВМ, позволяющая увеличить производительность системы ввода-вывода за счет уменьшения числа физических операций на дисках.

Программа управления кэшированием дисков (DCM) — специальный предпроцессор, управляющий всеми драйверами дисковых устройств.

Программа DCM контролирует все операции ввода-вывода на дисковых устройствах, что дает возможность уменьшить число физических операций за счет использования кэшовой области в оперативной памяти. В данной области хранятся копии логических блоков дисков. Это позволяет заменить передачу данных между диском и памятью на передачу «память — память». Такая замена осуществляется только для тех блоков, копии которых уже находятся в памяти и, следовательно, считается вероятностной функцией.

Программа DCM включается в ОСРВМ во время генераций системы и является частью полнофункциональной управляющей программы.

Возможность кэширования диска должна быть активизирована для каждого устройства отдельно. Пользователю предоставляются команды запуска, остановки, наблюдения или модификации кэширования дисков.

Кэширование данных диска может быть ориентировано на операции ввода-вывода определенного типа. При этом оно наиболее эффективно в случаях повторного чтения одних и тех же данных или их последовательного считывания небольшими записями.

Операции кэширования могут быть ориентированы на пять типов операций на дисках:

1) каталоговый ввод-вывод. По умолчанию данный тип активизирован. К каталоговому вводу-выводу относятся все операции, выполняемые файловым процессором с переключением контекста. Наиболее частые операции файлового процессора относятся к следующим файлам:

[0, 0] ITMAP. SYS
[0, 0] NDEXF. SYS
[0, 0] 00000. DIR

2) загрузка перекрытий. По умолчанию данный тип активизирован. Кэшируются операции ввода-вывода с кодами функций IO.LDO и IO.LOV. При этом анализируется длина запроса ввода-вывода и, если она оказывается слишком большой, кэширование не выполняется. Это позволяет избежать монополизации буфера кэширования операциями загрузки перекрытий;

3) ввод-вывод виртуальных блоков. По умолчанию данный тип активизирован. Кэшируются все операции ввода-вывода с кодами функций IO.RVB и IO.WVB. К операциям данного типа относятся также запросы файлового процессора, выполняемые без переключения контекста;

4) ввод-вывод логических блоков. По умолчанию данный тип запрещен. Логическим вводом-выводом блоков считаются все операции с кодами функций IO.RLB и IO.WLB (не включая запросов файлового процессора). Такие запросы выполняются программами, как правило, для специальных целей. Например, программа BRU выполняет чтение блоков копируемого диска согласно своим алгоритмам. В таких случаях кэширование увеличивает производительность весьма незначительно;

5) чтение виртуальных блоков с опережением. По умолчанию данный тип запрещен. После чтения последнего блока сегмента файла, находящегося в кэше, с помощью операций чтения виртуальных блоков система автоматически осуществляет считывание следующего сегмента без запроса программы пользователя. Максимальное число блоков, считываемых в кэш, равно числу блоков в сегменте или максимальному размеру сегмента. Операции кэширования данного типа имеют смысл в

случае интенсивного доступа к файлам с последовательной организацией.

Во время операций чтения, тип которых разрешен для кэширования, DCM перехватывает запросы ввода-вывода и проверяет наличие запрашиваемых данных в памяти. Если в памяти имеется их копия, DCM передает данные пользователю. Если в памяти нет запрашиваемых данных, DCM передает запрос соответствующему драйверу. После завершения физической операции ввода-вывода DCM копирует требуемую часть данных в буфер пользователя.

Во время операций записи, тип которых разрешен для кэширования, передаваемые данные копируются из буфера пользователя в соответствующий буфер кэша. Если такого буфера в кэше нет, запрос передается непосредственно в драйвер, а данные не кэшируются. После завершения операции копирования DCM завершает операцию ввода-вывода путем передачи пакета пользователя подпрограмме $IOFIN.

Для надежного отображения данных монтированных дисков в буферах кэша применяются специальные процедуры. В частности, для этого предназначена операция ввода-вывода с кодом IO.STC. В системе ОСРВМ код запроса IO.STC используется командами MOUNT и DMOUNT (с ключом /UNLOAD) для установки доступности тома. Когда программа управления кэшированием получает параметры монтирования, производится инициализация кэша. Далее DCM дублирует и диспетчеризует пакет пользователя, одновременно сохраняя оригинальный пакет. Код состояния завершения операции ввода-вывода возвращается через подпрограмму $IOFIN. Инициализация кэша заключается в очистке буферов, что предотвращает их применение после снятия диска. Если операция завершилась успешно, счетчик использования соответствующего района кэша увеличивается на единицу, что предотвращает его удаление. При получении пакета с запросом о демонтировании устройства выполняется его задержка до завершения переписи всех копий блоков на диск. После этого для данного устройства все операции кэширования запрещаются. Вслед за завершением переписи блоков из кэша на диск задержанный запрос на демонтирование передается в драйвер ввода-вывода. Условием активизации программы DCM является запрос к дисковому устройству массовой памяти (DV.MSD). Если устройство, к которому направляется запрос ввода-вывода, не является диском (или система кэширования не включена в систему), то осуществляется вызов драйвера через секцию инициализации. Такое подключение программы DCM в управляющую программу позволяет перехватывать все запросы ввода-вывода до их обработки секцией инициации драйвера и постановки в очередь устройства. Перед активизацией программы DCM необходимо установить районы кэширования и ряд связей в управляющей программе.

По умолчанию во время генерации ОСРВМ в командный файл SYSVMR.CMD включаются две следующие командные строки, позволяющие зафиксировать в образе системы задачу управления кэшированием:

```
INS   DCMEXP
FIX   DCMEXP/DIR
```

Программа DCM устанавливается в общую область директив.

Перед активизацией кэширования необходимо создать и сформатировать район данных кэша. Такой район создается системой автоматически, при этом пользователь может указать его имя. Когда устройство (устройства), связанное с районом данных кэша, завершает свою работу, район может быть переведен в неактивное состояние (но не удален) с помощью команд DISMOUNT или SET.

Район данных кэша состоит из сегментов. Они представляют собой физически последовательные блоки данных в памяти, которые соответствуют физически последовательным блокам на диске. Такие блоки содержат соответствующие номера логических блоков (LBN) на диске, копии которых находятся в памяти.

Блоки данных диска запоминаются в районе кэша во время операции чтения или в случае разрешения чтения с опережением. Если чтение с опережением разрешено, блоки данных, располагающиеся в том же сегменте, что и считываемые данные, также размещаются в районе кэша. При размещении в памяти копий блоков диска сохраняется информация об их логических номерах, что позволяет во время последующих операций ввода-вывода находить соответствующие данные.

Для поддержки района данных кэша необходимо зарезервировать адекватный объем памяти, позволяющий вместить копии дисковых блоков. Для наблюдения за состоянием района кэша можно воспользоваться командой SHOW или информацией, распечатываемой программой DCM.

В ОСРВМ, как правило, имеется один главный раздел для размещения различных объектов системы.

По умолчанию он называется GEN. Этот раздел может содержать образ задач общие области, разделы расширения управляющей программы. Для целей кэширования дисков необходим специальный район, содержащий буфера данных диска и ряд управляющих структур.

Районы кэширования могут быть индивидуальны для каждого устройства или разделяться между несколькими. При разрешении кэширования на устройстве по умолчанию создается район с именем CACHE. В случае создания района с другим именем он должен быть сформатирован специальным образом до его использования.

Создание и удаление района данных кэша выполняются с помощью специальных команд. Во время создания района производится его форматирование. После разрешения кэширования в расширенный блок управления устройства (UCBX) записывается адрес блока управлением раздела (PCB) района.

Чтобы предотвратить освобождение района, связанного с устройством, в поле P.RMCT устанавливается счетчик числа устройств, связанных с данным районом. Такой район не может быть удален до тех пор, пока поле P.RMCT не станет равным нулю. После того как все устройства завершат работу с районом кэширования, разрешается его удаление из системы.

Район данных кэша состоит из двух частей. В первой части района находится область пула, а во второй — область буферов блоков данных диска. Системы с совмещенным пространством инструкций и данных отображают пул через регистр 6 диспетчера памяти, а его длина ограничена 4 Ксловами. Системы с раздельным адресным пространством отображают пул через регистры 5 и 6, а длина ограничивается 8 Ксловами. Для работы с динамическим пулом используются стандартные процедуры управляющей программы $ALOCB (входная точка $ALOC1) и $DEACB (входная точка $DEAC1). Рабочие структуры данных, специфичные для работы системы кэширования, размещаются в этом отдельном динамическом пуле, что позволяет минимизировать загрузку системного пула. Остаток района кэша используется для буферизации блоков данных дисков и распределяется квантами по 256 слов на границе 32 слов.

Для управления кэшированием используются команды как DCL, так и MCR, которые позволяют:

•назначить кэширование для устройства;

•указать необходимый тип кэширования;

•следить за операциями кэширования;

«модифицировать параметры кэширования.

Команда MOUNT разрешает использование тома. Для того чтобы на выбранный диск назначить операции кэширования в команде MOUNT, необходимо использовать ключ /CACHE. Ф о р м а т :

MOUNT DDNN:метка-тома/[NO] CACHE [: (аргумент, аргумент, ...)]

где DDNN: — имя устройства, на котором монтируется том;

метка тома — метка, связанная с данным томом.

К л ю ч и команды MOUNT:

/CACHE — назначает кэширование для данного тома и позволяет задать различные параметры.

Параметры кэширования задаются с помощью аргументов в следующем виде:

CREATE '[: [район] [: главный-раздел] [: размер-кэша]]]]
REGION: имя
[NO] DIRECTORY [:размер-сегмента]
[NO] LOGICAL [:размер-сегмента]
[NO] OVERLAY [:размер-сегмента]
[NO] EAD_AHEAD [:размер-сегмента]
[NO] IRTUAL [:размер-сегмента]

Аргумент «размер-сегмента» указывает максимальную длину операции ввода-вывода, обрабатываемой системой кэширования. Размер сегмента задается в десятичном виде. Минимальное значение размера сегмента равно 1, максимальное—15. Размер сегмента может быть изменен с помощью команды SET;

/NOCACHE — отменяет кэширование для монтируемого устройства.

В ключе /CACHE может быть задано несколько аргументов:

CREATE [: [район] [:главный-раздел] [: [размер-кэша]]]] — создает район для кэширования и назначает его для указанного устройства. По умолчанию району присваивается имя CACHE, а в качестве главного раздела используется имя GEN. Размер кэша по умолчанию равен 100 (десятичным) блокам диска;

REGION: имя — назначает существующий район указанному устройству. Этим способом можно задать имя района, отличное от имени, принятого в системе по умолчанию (CACHE);

[NO] DIRECTORY[:размер-сегмента] — разрешает кэширование каталогового ввода-вывода. По умолчанию данный тип кэширования разрешен. NODIRECTORY — запрещает кэширование каталогового ввода-вывода. По умолчанию размер сегмента для данного типа кэширования равен 1;

[NO] LOGICAL[:размер-сегмента] — разрешает кэширование логического ввода-вывода блоков. NOLOGICAL — запрещает этот тип, По умолчанию используется NOLOGICAL, а размер сегмента равен 1;

[NO] OVERLAY[:размер-сегмента] — разрешает кэширование операций загрузки перекрытий. NOOVERLAY — запрещает этот тип. По умолчанию используется OVERLAY, а размер сегмента равен 4;

[NO] READ_AHEAD [:размер-сегмента] — разрешает чтение виртуальных блоков до того, как к ним будет получен запрос от программы пользователя. По умолчанию система использует NOREAD_AHEAD, запрещающий Чтение с опережением, и размер сегмента равным 5;

[NO] VIRTUAL [:размер-сегмента] — разрешает кэширование виртуальных блоков, а NOVIRTUAL запрещает данный тип. По умолчанию используется VIRTUAL с размером сегмента, равным 5.

Команда DCL SHOW CACHE позволяет вывести на терминал информацию о работе системы кэширования с указанным диском. После инициализации кэширования данных диска этой командой можно воспользоваться для наблюдения за работой и производительностью системы. Ф о р м а т :

SHOW CACHE [/DEVICE: = DDN:] [/REGION = имя] [/RATE: N]

где /DEVICE = DDN:— специфицирует устройство, работа которого должна быть отображена;
/REGION = имя — специфицирует район, работа которого должна быть отображена;
/RATE: N — задает частоту обновления информации на терминале. По умолчанию равно 1 с.

Команда DCL SHOW DEVICE. С ее помощью после инициализации командами MOUNT или SET кэширования данных диска можно распечатать список устройств, участвующих в кэшировании. Если необходима информация о каком-либо конкретном устройстве, следует задать его имя. Ф о р м а т :

SHOW DEVICE: [DDNN:l/[NO] CACHE

где DDNN: — специфицирует устройство, о состоянии которого необходимо отобразить информацию;
/CACHE — приводит к распечатке информации о кэшируемых дисковых устройствах;
/NOCACHE — приводит к распечатке информации о некэшируемых в данный момент дисковых устройствах.

Команда SET DEVICE позволяет назначить или изменить параметры кэширования для устройства. С помощью данной команды имеется возможность изменить работу системы и ряд ее параметров как для отдельного устройства, так и для их набора. При этом параметры района не могут быть изменены, пока он используется для кэширования. Ф о р м а т :

SET DEVICE DDNN: /[NO] CACHE[:(аргумент, аргумент, ...)]

где DDNN: — имя модифицируемого устройства;
/CACHE — изменяет работу для указанного устройства. Разрешается применять следующие аргументы:

CREATE [:район] [:[главный-раздел] [:[размер-кэша]]]
REGION :имя
[NO] RECTORY [:размер-сегмента]
[NO] OVERLAY [:размер-сегмента]
[NO] READ_AHEAD [:размер-сегмента]
[NO] VIRTUAL [:размер-сегмента]

Аргумент размер-сегмента указывает максимальную длину операции ввода-вывода, обрабатываемую системой кэширования. Размер сегмента задается в десятичном виде. Минимальное значение размера сегмента равно 1, максимальное— 15. Размер сегмента может быть изменен с помощью команды SET;

/NOCACHE — останавливает кэширование для указанного устройства.

Команды MCR позволяют выполнить следующие действия:
назначить кэширование для устройства;
указать необходимый тип кэширования;
следить за операциями кэширования;
модифицировать параметры кэширования.
Формат команды MOUNT MCR и действия, выполняемые ею, совпадают с командой MOUNT DCL.

После запуска кэширования данных диска имеется возможность воспользоваться командой RMD для анализа состояния системы.

Команда RMD позволяет отображать на дисплее различные динамические страницы, содержащие информацию о системе. В частности, команда "RMD C" выводит страницу C, содержащую статистические данные о некотором районе кэша. Анализ данных, приведенных в странице, позволяет более обоснованно модифицировать системные переменные с целью улучшения производительности. Ф о р м а т :

RMD C [,REGION = район] [,RATE = NN]

где REGION — определяет имя отображаемого района. По умолчанию используется CACHE;
RATE — специфицирует период обновления страницы в секундах. По умолчанию используется 1 с.

Более подробные данные о состоянии района кэша можно получить в странице D команды RMD. Ф о р м а т :

RMD D [,DEVICE1=DDNN:] [,RATE = NN]

где DEVICE1 — задает имя устройства. По умолчанию SY:;

RATE — период обновления страницы. По умолчанию используется 1 с.

Поля данных в примере, представленном на рис. 8, имеют следующие значения:

OCPBM V1.0	CACHE STATISTICS (DETAILED)			21-DEC-87		9:15 00
DEVICE NAME: SYO:	1) REGION NAME:	CACHE REGION SIZE	5000			
2) CACHE STATUS: ACTIVE	3) REOUTESTS BEING CACHED: DIR, OVP, VIR					
	VIRTUAL	READAHEAD	DIRECTORY	LOGICAL	OVERLAY	TOTAL
4) READS	77515	0%	21755	0%	16919	116189
READ HIT RATE	73%	0%	72%	0%	49%	67
5) READ LOAD RATE	22%	0%	17%	0%	4%	11
READ OVERLAP	0%	0%	2%	0 %	0%	
EXTENT TOO BIG	3%	0%	0%	0%	2%	
6) MAX EXTENT SIZE	5.	5.	1.	1.	4.	
7) WRITES	12966.		12507.	105.		255.
8) WRITE HIT RATE	48%		91%	0 %		
WRITE OVERLAP	0%		0%	5%		
9) TOTAL I/O	90471.	0.	34262.	105.	16919.	141775
10) PRIMARY POOL	ALLOCATION FAILURE RATE (AS A% OF TOTAL l/OS)				0%	
11) CACHE POOL	ALLOCATION FAILURE RATE (AS A% OF TOTAL l/OS)				0%	
12) READ LOAD FAILURE RATE (AS a% OF CACHE LOAD l/OS):					0%	
13) DEFERRED WRITE RATE (AS a% OF TOTAL WRITE l/OS):					0%	

Рис. 8. Пример страницы D, отображаемой с помощью команды RMD

1) — имя района и его размер;

2) — поле состояния кэша, которое может принимать следующие значения:

ACTIVE — устройство кэшируется с использованием данного района;

ENABLED — кэширование устройства разрешено и будет активизировано после выполнения команды MOUNT (даже если в ней не указан ключ /CACHE);

3) — тип запросов, для которых производится кэширование:

VIRTUAL (VIR), READ__AHEAD (RDH), DIRECTORY (DIR), LOGICAL (LOG), OVERLAY (OVR);

4) — число запросов чтения для каждого типа операций, а также их общая сумма;

5) — данные четыре строки содержат процентное количество операций чтения из кэша, количество операций физического чтения, чтения с перекрытием и чтение с превышением размера сегмента. Каждая из этих строк содержит счетчики отдельно по каждому типу операций;

6) — максимальный размер сегмента по каждому типу операций;

7) — число операций записи каждого типа в процентах, а также их общая сумма. Поля READAHEAD и OVERLAY для операций записи отсутствуют;

8) — число операций записи, удовлетворенных в кэше, и число операций записи с перекрытием в процентах от общего числа операций записи;

9) — общее число операций ввода-вывода каждого типа и общее число операций. Последние четыре строки указывают число операций ввода-вывода, не вошедших ни в один из типов. Все операции ввода-вывода требуют, чтобы пакет запроса был размещен в первичном пуле;

10)— процент запросов ввода-вывода, для которых получен отказ при резервировании первичного пула;

11)— процент запросов чтения, для которых получен отказ при резервировании пула в кэше;

12)— процент запросов чтения данных из диска в кэш, при выполнении которых произошла ошибка;

13)— процент операций записи с задержкой, выполненных после завершения запроса пользователя.

Команда DEV отображает информацию о различных устройствах. Если имя устройства не задано, выводится информация о всех устройствах, известных системе. В случае отсутствия номера устройства отображается информация о всех устройствах с данным именем. Полное указание номера и имени приводит к отображению всей известной информации о данном конкретном устройстве. Ф о р м а т :

DEV [DD.[NN]:]

где DD: — имя устройства;

NN — номер устройства.

Команда SET обеспечивает отображение или модификацию различных параметров системы, в том числе параметров, относящихся к кэшированию.

Команда SET/[NO]CACHE позволяет распечатать или модифицировать параметры кэширования для конкретного устройства или их группы. Размер района может быть указан только во время его создания. Имя района может быть изменено только после завершения всех операций ввода-вывода,

связанных с данным районом. Ф о р м а т :

SET /[NO] CACHE [:DDNN: [(аргумент, ...)]]

К л ю ч и команды SET:

/CACHE — если не указано устройство, ключ /CACHE приводит к распечатке устройств, для которых разрешено кэширование. Если устройство указано, разрешает кэширование и позволяет модифицировать ряд параметров;

/NOCACHE — распечатывает список устройств, для которых кэширование запрещено. Возможно указание конкретного устройства, но модификация не разрешена;

DDNN: — имя и номер устройства.

А р г у м е н т ы :

CREATE [:[район] [:[главный-раздел] [:[размер]]]]
REGION: имя
[NO] DIRECTORY [:размер-сегмента]
[NO] LOGICAL [:размер-сегмента]
[NO] OVERLAY [:размер-сегмента]
[NO] READ_AHEAD [:размер-сегмента]
[NO] VIRTUAL [:размер-сегмента]

6.8. ПРОГРАММЫ ЭМУЛЯЦИИ ТЕРМИНАЛА И ПЕРЕДАЧИ ФАЙЛОВ

Программы эмуляции терминала (DTE) и передачи файлов (MFT) предназначены для обмена данными между двумя системами, объединенными линиями связи. В качестве удаленных систем могут выступать комплексы СМ ЭВМ (СМ 1420, СМ 1425, СМ1700), работающие под управлением ОСРВ, ОСРВМ, МОСВП. С помощью программы DTE пользователь локального терминала может зарегистрироваться в удаленной системе, которая для него становится главной системой.

Во время сеанса работы между системами могут использоваться команды удаленной системы, а также выполняться программой MFT передача файлов.

Для обеспечения работоспособности системы необходимо:

•установить связь между системами;

•в локальной системе установить программу DTE, в удаленной — программу MFT.

Программы DTE и MFT не обеспечивают и не заменяют функций сети ЭВМ, в частности пакета СЕТЬ-СММ. Основное назначение программ эмуляции состоит в возможности взаимодействия двух систем простыми средствами без сложной процедуры генерации. При этом набор предоставляемых возможностей весьма ограничен.

Физическую связь между двумя (возможно, неодинаковыми) системами наиболее просто установить с помощью стандартных стыков последовательной передачи данных. Для работы по выделенным каналам связи можно рекомендовать интерфейс для радиального подключения устройств с последовательной передачей информации — ИРПС (известный под названием «20 мА токовая петля»). Для передачи на большие расстояния следует устанавливать связь через стык С2. Стык С2 можно использовать для локальной связи по выделенным линиям на расстоянии до 15 м без модемов. При более длинных линиях следует применять модемы. Данные стандарты передачи данных, поддерживаемые аппаратурой вычислительных комплексов СМ ЭВМ, реализованы в одноканальных адаптерах, а также в мультиплексорах передачи данных.

Необходимая аппаратура и программы DTE и MFT должны поддерживаться с помощью терминального драйвера.

На локальной системе терминальная линия связи, используемая для передачи данных, должна быть установлена в режиме подчинения (SLAVE) и подавления эхо-контроля (NOECHO). Это предотвращает передачу по линии связи циклических эхо-сигналов, возбуждаемых электрическими шумами от внешних источников помех.

Скорость передачи данных между системами зависит от конкретных характеристик канала связи. На выбор скорости влияют следующие факторы:

•тип процессора;

•тип интерфейса;

•нагрузка системы;

•расстояние между системами;

•возможности драйвера терминала;

•уровень шума на линии.

Рекомендуемая скорость передачи, при которой обеспечивается приемлемый режим работы, равняется 2400 бит/с.

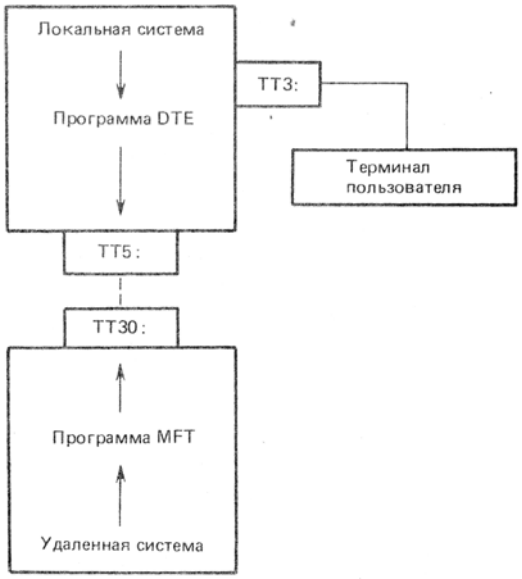

Рис. 9. Пример установки связи между двумя системами

Пользователь локальной системы, представленной на рис. 9, использует терминальный порт с именем TT3:. При помощи программы DTE терминал TT3: логически подключается к порту TT5:, а он физически соединен с портом TT30: удаленной системы. Чтобы исключить влияние шумов на линии конфигурации, приведенной на рис. 9, следует с терминала пользователя (или с консольного терминала) ввести следующую команду:

SET TERMINAL TT5:/SLAVE/NOECHO

Эмуляция удаленного терминала и передача файлов поддерживаются двумя программами, одна из которых расположена в локальной системе, а другая — в удаленной. Программа DTE располагается в локальной системе и обеспечивает эмуляцию терминала. Эмуляция терминала означает такое логическое подключение к удаленной системе, которое обеспечивает вход локального пользователя в удаленную систему.

Программа MFT располагается в удаленной системе и обеспечивает взаимодействие с программой DTE во время операций передачи файлов. Таким образом, DTE и MFT являются взаимосвязанными программами, которые работают в физически разных системах.

После объединения машин с помощью физической линии и установки программ DTE и MFT в соответствующих системах можно активизировать эмуляцию терминала. Для этого используется команда SET на локальном терминале. Ф о р м а т :

SET HOST/DTE TTNN: /MUTE

где TTNN: — имя территориального порта, используемого для выхода в удаленную систему. Ключ /MUTE обеспечивает установку порта TTNN: в режимы SLAVE и NOECHO после выхода из программы DTE.

Для конфигурации, приведенной на рис. 9, данная команда будет иметь следующий вид:

SET HOST/DTE TT5: /MUTE

Команда DCL запускает задачу DTE, которая активизирует эмуляцию терминала путем логического объединения с портом TT5:. После завершения эмуляции порт TT5: будет установлен в режимы SLAVE и NOECHO.

Данная команда подсоединяет локальный терминал к удаленной системе, и затем можно выдавать команды регистрации в этой системе, а также запускать передачу файлов.

Для завершения эмуляции прежде всего необходимо выйти из удаленной системы с помощью команды DCL LOGOUT и завершить программу DTE вводом управляющего символа УС/P (или CTRL/P).

Индикацией завершения эмуляции является следующее сообщение, выдаваемое на локальный терминал:

%DTE-S-EMUEXIT, EMULATION EXITING ... PLEASE WAIT

(Эмуляция завершается... Ждите)

После этого локальный терминал продолжает обслуживаться интерпретатором команд локальной

машины.

П р и м е ч а н и е . Необходимость выхода из удаленной системы перед возвратом в локальную систему с помощью управляющего символа УС/Р (CTRL/P) обусловливается следующим:

1) если в удаленную систему не была выдана команда LOGOUT, другой пользователь сможет установить эмуляцию и воспользоваться ресурсами, выделенными для пользователя, установившего связь первым;

2) если программа DTE используется с консольного терминала, прежде всего необходимо отключить отладчик пульта, который также использует управляющий символ УС/Р (CTRL/P).

Программа MFT используется для управления передачей файлов в режиме эмуляции терминала. В программе DTE поддерживаются операции по передаче файлов, запрашиваемые программой MFT. Передача файлов осуществляется с помощью алгоритмов обнаружения и корректировки ошибок, что обеспечивает целостность передаваемых данных.

Скорость передачи файлов зависит от типа процессора и интерфейсов, загрузки системы и режимов работы терминального порта. В частности, поддержка буферизации ввода в терминальном драйвере обеспечивает более высокую скорость передачи.

Во время передачи файлов локальный терминал заблокирован. Это позволяет избежать конфликтных ситуаций. После завершения операции работа терминала возобновляется.

Команды, используемые для активизации передачи файлов, зависят от возможностей удаленной операционной системы. Если удаленная система работает под управлением ОСРВ или ОСРВМ и использует интерпретатор команд оператора MCR, запуск программы MFT осуществляется следующей командой:

```
>MFT
```

Удаленная система должна ответить следующим образом:

```
MFT>
```

Далее можно запустить копирование файлов с одной системы на другую в любую сторону. Для этого используется следующая команда:

```
MFT>OFC [/REM] = IFC [/REM]
```

где OFC — спецификация выводного файла;
IFC — спецификация вводного файла;
/REM — означает, что данный файл находится (или должен быть создан) на локальной системе.

Локальная система по отношению к программе MFT является удаленной. Например, если копируется файл из удаленной системы в локальную, используется команда

```
MFT>MESSAGE. TXT/REM = HOST. TXT
```

При передаче файла в обратном направлении используется команда

```
MFT>MESSAGE. TXT = MICRO. TXT/REM
```

Файл удаляется командой

```
MFT>DELFIL  [/ключ]
```

где DELFIL — спецификация удаляемого файла.

К л ю ч и :

/DE — указывает операцию удаления файла;
/REM — определяет систему, на которой должен быть удален файл. В спецификации файла при команде удаления использования звездочки * не разрешается.

П р и м е р :

```
>SET HOST/DTE TT5:
>LOGIN
   >ACCOUNT OR NAME:AAAA
   >PASSWORD:BBBB
         .
         .
         .
   >MFT
   MFT>MESSAGE. TXT = MICRO. TXT/REM
   %DTE-S-MFTINIT, FILE OPERATION INITIATED
            (передача файла начата)
   %DTE-S-MFTCOMP, FILE OPERATION COMPLETE
```

(передача файла завершена)

MFT>

где AAAA — имя пользователя;
BBBB — пароль.

В данном примере оператор осуществляет передачу файла MICRO. TXT из локальной системы в файл MESSAGE. TXT в удаленную систему. Во время передачи программа MFT сообщает о начале и конце операции.

Работа завершается следующим образом:

```
>LOGOUT
        .
        .
        .
CTRL/P
%DTE-D-EMUEXIT,  EMULATION  EXITING ... PLEASE WAIT
                (Эмуляция завершается... Ждите)
```

Если удаленная система обслуживается диалоговым командным языком (DCL), для передачи файлов используются следующие команды:

COPY IFC [/REM] OFC [/REM] для передачи файлов

DELETE FC [/REM]; для удаления файлов

где IFC — спецификация вводного файла;
OFC — спецификация выводного файла;
/REM [OTE] — используется для указания, что данный файл находится (или должен быть создан) на локальной системе. Отсутствие ключа /REM означает, что файл находится на удаленной системе;
FC — спецификация удаляемого файла.

П р и м е р ы :

> COPY INFILE.DAT OUTFILE. DAT/REMOTE

Переписывается файл INFILE. DAT из удаленной системы в файл OUTFILE. DAT локальной системы.

> COPY INFILE. DAT/REMOTE OUTFILE.DAT

Переписывается файл INFILE. DAT из локальной системы в файл OUTFILE. DAT удаленной системы.

>DELETE FILE. DAT/REMOTE

Удаляется файл FILE. DAT локальной системы.

6.9. ДИСПЕТЧЕР НЕУСТАНОВЛЕННЫХ ЗАДАЧ

Функции диспетчера неустановленных задач ОСРВМ выполняет программа TDX. Любая задача с именем ...CA. рассматривается в ОСРВМ как задача, перехватывающая команды, не распознанные в MCR. Программа TDX устанавливается в ОСРВМ следующим образом:

>INS $TDX/TASK = ...CA.

Если программа MCR не распознала команду оператора, она осуществляет поиск задачи с именем ...CA. и передает ей командную строку. Программа TDX проверяет полученную команду и, если она распознана, передает программе MCR расширенную строку в соответствии со своей внутренней таблицей.

Команды программы TDX представляют собой сокращения ряда команд ОСРВМ и их ключей. В табл. 6.2 представлены сокращенные команды программы TDX, соответствующие командные строки MCR и описание значения команд.

Таблица 6.2

Команда TDX	Команда MCR	Значение команды
ATS	ACT /ALL	Распечатка имен всех активных задач в системе
ATS TTNN:	ACT /TERM = TTNN:	Распечатка имен всех активных задач для данного терминала
CHD	SET /DEF	Распечатка текущего каталога по умолчанию для терминала TI:
CHD G M	SET /DEF=[G, M]	В режиме именованных каталогов изменяет текущий каталог по умолчанию на заданный G и M. В режиме неименованных каталогов изменяет текущий каталог по умолчанию и, если привилегированный терминал, UIC защиты
CHU	SET/UIC	Распечатывает UIC защиты для терминала TI: и, если режим неименованных каталогов, — UFD по умолчанию

Продолжение

Команда TDX	Команда MCR	Значение команды
CHU G M	SET /UIC = [G, M]	В режиме именованных каталогов изменяет UIC защиты на UIC, заданный G и M (привилегированная команда), в режиме неименованных каталогов изменяет каталог по умолчанию и, если терминал привилегированный, — UIC защиты
CLR	–	Очищает экран терминала и устанавливает курсор в позицию 0,0. Возвращает состояние EX $ SUC, если это видеотерминал, и EX $ WAR, если терминал на основе печатающего устройства
CRE FILE	PIP FILE = TI:	Создает новый файл
CVT VAL	–	Вычисляет арифметическое выражение, преобразует его в различные форматы и выводит на терминал, CVT позволяет задавать значение VAL в различных форматах: восьмеричном NNN или NN,NN десятичном NNN. или NN,NN шестнадцатеричном $ NNNN RADIX-50 %CCC символьном ’C или ”CC в виде арифметических выражений с использованием знаков арифметических операций +, —, /, ∗ и < CVT практически можно использовать в качестве калькулятора и для преобразования форматов данных
DEL FILE	PIP FILE/DE	Удаляет файл
DIR [FILE]	PIP [FILE]/LI	Распечатывает каталог на терминал
DLG	DEV/LOG	Распечатывает информацию о всех терминалах, зарегистрированных в системе
DLN	NCP SHOW KNOWN NODES	Распечатывает имена всех узлов, известных в сети
FRE	PIP/FR	Распечатывает количество свободного пространства на устройстве SY:, длину наибольшего последовательного свободного участка, число свободных заголовков файла и число использованных заголовков
FRE DDU:	PIP DDU: /FR	Распечатывает ту же информацию, что и FRE, для указанного устройства DDU:
PUR FILE	PIP FILE/PU	Удаляет все, кроме последней версии файла
SHQ	QUE /LI	Распечатывает информацию о состоянии всех очередей системной печати
SYS	SET/SYSUIC	Распечатывает текущий системный UIC
TYP FILE	PIP TI: =FILE	Распечатывает на терминале файл

Если введенная с терминала команда не соответствует приведенным в табл. 6.2, программа TDX пытается выполнить одно из двух действий:

1) установить, запустить и удалить задачу из системы. Для этого используется команда MCR следующего вида:

RUN $XXX/TASK = XXXTNN/CMD = «параметры...»

где XXX — первые три буквы введенной команды;
параметры — параметры во введенной команде.

Если желательно использовать данную возможность программы TDX, необходимо в командном файле LOGIN.CMD добавить следующую строку:

ASN SY: = ZZ1:

Длина команды не должна превышать 39 символов;

2) запустить выполнение косвенного командного файла одним из следующих способов:

@ SY: [LOGINUIC]XXX.CMD
@ LB:[LIBUIC]XXX.CMD
@ SY: [LOGINUIC]CATCHALL.CMD
@ LB: [LIBUIC]CATCHALL.CMD

где XXX — первые три буквы введенной команды;
LOGINUIC — код идентификации пользователя по умолчанию;
L1BUIC — код идентификации библиотеки по умолчанию.

Программа TDX осуществляет поиск указанных файлов в приведенном порядке. После обнаружения файла поиск заканчивается. Данный способ позволяет выполнять различные функции для установки задач, распечатки справочной информации или сообщений об ошибках. Для этого в командном файле LOGIN.CMD следует добавить строку:

```
ASNN SY: = ZZ2:
```

Выбор возможности зависит от назначения устройств ZZ1: и ZZ2:. При этом назначение ZZ1: более приоритетно по сравнению с ZZ2:. Если не выполнено ни одно из назначений, программа TDX не пытается выполнить приведенные действия.

Выбрать режим работы TDX можно путем глобального назначения в стартовом командном файле:

```
ASN SY: = ZZN:/GBL
```

где N — равняется 1 и 2 в зависимости от желаемого режима работы.

Если все попытки программы TDX распознать введенную команду окончились неуспешно, выдается сообщение

```
MCR— — TASK NOT IN SYSTEM
(задача в системе отсутствует)
```

6.10. УПРАВЛЕНИЕ ПАКЕТНЫМ РЕЖИМОМ И ОБРАБОТКОЙ ОЧЕРЕДЕЙ

Одним из возможных режимов работы пользователя с ОСРВМ является режим пакетной обработки. В этом случае пользователь строит запрос на обработку в виде специальной конструкции, называемой *пакетным заданием.*

Пакетное задание имитирует интерактивный сеанс работы пользователя на терминале без специального вмешательства с его стороны. С помощью специальной команды оператора пакетные задания направляются в очередь пакетной обработки, откуда выбираются для выполнения.

Аналогичный механизм очередей реализован в ОСРВМ и для управления выводом на печатающие устройства.

Очередями вывода на печать и очередями пакетных заданий управляет системная обслуживающая программа, называемая *диспетчером очередей* (QMG).

В функции диспетчера очередей входит распределение заданий вывода на печать и пакетных заданий в соответствующие очереди, откуда они выбираются на обработку. Диспетчер QMG осуществляет полный контроль за очередями, включая возможность удаления задания из очереди или задержки их для более поздней обработки. Очереди размещаются диспетчером QMG на библиотечном устройстве в файле [1, 7] QUEUE.SYS, поэтому в случае аварийного завершения работы системы задания в очередях не теряются.

Число очередей и соответствующие устройства вывода для них (в случае вывода на печать) выбираются при генерации системы. Одна очередь может назначаться для нескольких устройств или вовсе не иметь связанных с ней устройств. В свою очередь с устройством может быть связано несколько очередей.

При создании очередей им назначаются определенные атрибуты, в частности имена, которые будут использоваться в дальнейшем. По умолчанию вывод на печать направляется в очередь с именем PRINT, а пакетное задание — в очередь с именем BATCH.

Пользователь передает задание QMG по команде PRINT в DCL или в MCR — в случае вывода на печать или SUBMIT — в случае пакетной обработки.

6.10.1. ОПИСАНИЕ КОМАНДЫ PRINT

Устройства вывода — как правило, печатающие устройства, которые контролируются QMG — называются *устройствами спулинга.* QMG управляет распечаткой файлов с помощью специальных программ, называемых процессорами вывода на печать. Такие процессоры имеются в системе для каждого устройства спулинга. Ф о р м а т команды:

```
DCL>PR1NT [/клк] спец.файла [/клф]
MCR>PRI [[имяо:] [имяз] [/клз]=] спец.файла [/клф]
```

где /клк — один или несколько ключей команды (DCL);
спец.файла — одна или несколько спецификаций файлов. По умолчанию тип файла— LST;
/клф — ключи, относящиеся к файлу;
/клз — ключи заданий (MCR);

/имяо — имя очереди заданий на печать;
/имяз — имя задания на печать.

К л ю ч и команды и заданий:

Ключи команды (DCL)	**Ключи заданий (MCR)**
/JOBCOUNT: N	/CO: N
/QUEUE :имяо	имяо:
/UPPERCASE	/NOLOW
/LOWERCASE	/LOW

Ключи команды (DCL)	**Ключи заданий (MCR)**
/[NO] HOLD	/[NO] HO
/PAGE_COUNT: N	/PA:N
/NAME: имяз	имяз =
/PRIORITY: N	/PRIO:N
/FORM: N	/FO:N
/LENGTH: N	/LE:N
/[NO] RESTART	/[NO] RES
/[NO] FLAG_PAGE	/[NO] FL
/AFTER: (DD—MMM—YY HH:MM) (MMM/DD/YY HH:MM)	/AF:HH:MM:DD — MMM — YY
/DEVICE: DDNN:	имяо:
/[NO] JOB_PAGE	/NOJO
JOB_PAGE	

где /JOBCOUNT: N

/CO: N — указывает число копий (N) задания на печать. Задание на печать состоит из одного или нескольких файлов, которые распечатываются в том порядке, в котором они перечислены;

/QUEUE: имяо

имяо — указывает имя очереди на печать, в которой должно размещаться задание. По умолчанию имя очереди PRINT;

/UPPERCASE

/NOLOW — указывает, что задание на печать может быть распечатано на устройстве без набора символов нижнего регистра. Этот ключ используется по умолчанию;

/LOWERCASE

/LOW — указывает, что задание на печать должно быть распечатано на устройстве с набором символов нижнего регистра;

/[NO] HOLD

/[NO] HO — указывает, что задание должно быть задержано в очереди и недоступно для печати. Вывести задание из этого состояния можно с помощью команды RELEASE/JOB (QUE/RE);

/PAGE_COUNT: N

/PA: N — устанавливает лимит (от 1 до 65535) количества страниц которое может вывести на печать задание. Если данный ключ не указан, количество страниц не ограничено;

/NAME: имяз

имяз = — указывает имя задания на печать длиной не более девяти алфавитно-цифровых символов. Если имя задания указано, то имя распечатывается в странице заголовка задания в начале вывода на печать. В противном случае для имени задания используется имя первого файла в задании;

/PRIORITY: N

/PRIO: N — устанавливает приоритет (N) задания на печать в очереди. Для непривилегированных пользователей приоритет может иметь значение от 0 до 150, а для привилегированных пользователей — до 250. По умолчанию N = 50. QMG сначала обрабатывает задание с наивысшим приоритетом. Если два задания имеют один и тот же приоритет, то сначала обрабатывается то задание, которое поставлено в очередь раньше;

/FORM: N

/FO: N — указывает, в каком формате должно распечатываться задание. Значение N может быть от 0 до 255. По умолчанию N = 0.

/LENGTH: N

/LE: N — устанавливает размер логической страницы. N может быть любым числом от 0 до 255. По умолчанию N = 0. Если пользователь устанавливает размер страницы, то перевод формата, т. е. прогон страницы, генерируется автоматически после N строк. Если устанавливается N = 0 (по умолчанию), то автоматический прогон страницы не выполняется;

/[NO] RESTART

/[NO] RES — если указан данный ключ, то выполнение задания повторяется сначала после остановки или аварийного

завершения задания. Значение по умолчанию — /NORESTART (/NORES). В этом случае, если задание было прервано, например, по причине отказа печатающего устройства, его обработка продолжается с прерванного места. Таким образом, задание всегда будет обработано полностью;

/[NO] FLAG_PAGE

/[NO] FL — добавляет страницу заголовка файла для каждого файла в задании вывода на печать. По умолчанию /NOFLAG_PAGE (/NOFL). В этом случае распечатываются только страницы заголовка задания, а файлы а задании распечатываются без страниц заголовков файлов;

/AFTER: (DD — MMM — YY HH:MM)
(MMM/DD/YY HH:MM)

/AF:HH:MM:DD — MM — YY — задания блокируются в очереди до указанного времени: HH — часы, MM — минуты, DD — день, MMM — месяц, YY — год. В зависимости от состояния очереди на печать в установленный момент времени задание может выполняться сразу или позднее, если в очереди имеются другие задания впереди данного. Время и дата указываются в том же формате, который указан в командах SET TIME (DCL) и TIM (MCR).

Если не указан ключ /AFTER (/AF), то задание немедленно получает доступ на печать;

/DEVICE: DDNN:

имяо: — назначает устройство вывода. Устройством вывода на печать по умолчанию являются устройства, обслуживающие очередь PRINT;

/[NO] JOB__PAGE

/[NO] JO — управляет выводом на печать страницы заголовка задания. По умолчанию — /JOB_PAGE (/JO).

Если пользователь указывает ключ, относящийся к файлам, в команде PRINT с использованием MCR, то этот ключ относится ко всем файлам, указанным в командной строке после файла, в котором указан ключ. Однако для любого файла это правило можно обойти, используя ключ, относящийся к конкретному файлу.

/COPIES: N

/CO: N — указывает количество (N) копий вывода на печать соответствующего файла в задании. По умолчанию N = 1;
/[NO] DELETE

/[NO] DE — указывает, что файлы после вывода их на печать должны быть удалены. По умолчанию —/NODELETE;

/ [NO] TRANSFER

[NO] TR — разрешает копирование файла с личного диска. По умолчанию — /TRANSFER.

Использование команды PRINT можно проиллюстрировать следующими примерами.

П р и м е р 1.

```
DCL>PRINT JOHANNA.TXT
MCR>PRI JOHANNA.TXT
```

Выводится на печать файл JOHANNA.TXT.

П р и м е р 2.

```
DCL>PRINT/NOJOB_PAGE KIRSTEN.TXT
MCR>PRI NOJO = KIRSTEN.TXT
```

Выводится на печать файл KIRSTEN. TXT без страницы заголовка задания.

П р и м е р 3.

```
DCL>PRINT/JOBCOUNT:4 JOHANNA. TXT
MCR>PRI/CO:4 = JOHANNA.TXT
```

Выводятся на печать четыре копии задания JOHANNA, каждая из которых включает одну копию файла JOHANNA. TXT. Копии распечатываются друг за другом со страницей заголовка задания, которая печатается перед каждой копией.

П р и м е р 4.

```
DCL>PRINT/LENGTH:55 DAD. MAC
MCR>PRI/LE:55 = DAD. MAC
```

Выводится на печать одна копия исходного файла программы DAD. MAC. Размер логической печатной страницы задан длиной 55 строк. После распечатки каждых 55 строк (одной логической страницы) печатающее устройство прогоняет бумагу на начало следующей физической страницы. Если внутри выводного файла встретится код перевода формата, то печатающее устройство также будет прогонять бумагу на начало следующей физической страницы.

П р и м е р 5.

```
DCL>PR1NT/NAME: SAILOR/JOBCOUNT: 2/DELETE JIMBO. DOC. — MOUNT/COPIES: 3. QUACK
```

MCR>PRI SAILOR/CO:2=JIMBO. DOC/DE,MOUNT/CO:3; QUACK/CO:l

Выводятся на печать две копии задания с именем SAILOR. Каждая копия задания на печать содержит одну копию файла JIMBO.DOC, три копии файла MOUNT. DOC и одну копию файла QUACK. DOC. Все файлы после вывода на печать удаляются.

Необходимо учитывать различие между ключом /JOBCOUNT:2 (/CO:2) для команды PRINT и ключом /C0P1ES:3 (/CO:3) для файла MOUNT. DOC.

П р и м е р 6 .

DCL> PRINT/AFTER : (12 : 15 10 — AUG — 87) JIMBO.TXT
MCR>PRI/AF : 12 : 15 : 10 — AUG — 87 = JIMBO. TXT

Блокируются задания в очереди до назначенной даты и времени.

6.10.2. ПАКЕТНАЯ ОБРАБОТКА

В системе ОСРВМ реализованы два типа пакетных заданий:

•пакетные задания пользователя;

•пакетные задания диспетчера QMG.

Пакетное задание пользователя является файлом, который создается пользователем и состоит из специальных пакетных команд, команд оператора, а также данных. Информация, содержащаяся в пакетном задании пользователя, должна соответствовать полному интерактивному сеансу на терминале. Пакетное задание должно обеспечить свою регистрацию в системе, управление операциями, а также выполнять выход из системы.

Пакетное задание пользователя обеспечивает почти все функции, допустимые при работе непосредственно с терминала, включая трансляцию, построение или отладку задач. Если пакетное задание пользователя передается для пакетной обработки, то оно формируется как пакетное задание QMG.

Как упоминалось выше, для передачи пакетного задания пользователя используется команда SUBMIT. В одной командной строке SUBMIT можно передавать несколько пакетных заданий пользователя для пакетной обработки. Несколько пакетных заданий пользователя называются *пакетным заданием QMG.*

Команды в пакетных заданиях передаются в ОСРВМ с помощью задач, которые называются процессорами пакетной обработки. В одной системе может содержаться до 16 таких процессоров и пакетных очередей.

Команда SUMBIT помещает файл пакетного задания пользователя в очередь к процессору пакетной обработки на выполнение. Ф о р м а т :

DCL> SUBMIT [/ключ [и]] файл[ы] [ключ [и] файла [ов]]
MCR>SUB [имя__очереди:] [имя задания] [/ключ[и] _задания]] = файл[ы] [ключ [и]_файла [ов]]

К л ю ч и команды и заданий

Ключи команд (DCL)
/AFTER: (мм/дд
/гг чч:мм)
(дд-ммм-гг чч:мм)
/[NO] HOLD
/NAME: имя_задания
/[NO] RESTART
/PRIORITY: N
/QUEUE; имя_очереди

Ключи файлов (DCL)

/[NO) DELETE /[NO] TRANSFER

Ключи файлов регистрации (DCL)

/NAME: имя_задания
/[NO] LOGFILE
/[NO] PRINT
/PRINT: очередь_на_печать

Ключи задания (MCR)
/AF:чч:мм:дд-ммм-гг
/[NO] HO
имя_задания =
/[NO] RE
/PRIO: N
имя_очереди:

Ключи файлов (MCR)

/[NO] DEL /[NO] TR

Ключи файлов регистрации (MCR)

имя_задания =
/[NO] LO
/[NO] PRIN
/PRIN: очередь_на_печать

Тип файла по умолчанию для команды SUBMIT — .CMD. Если необходимо повторно запустить задание с начала, то используется ключ /RESTART (/RE).

Для ключа /AFTER: (/AF:) в поле даты по умолчанию устанавливается текущая дата.

Использование команды SUBMIT можно проиллюстрировать следующими примерами.

П р и м е р 1 .

```
DCL>SUBMIT/PRIORITY: 150 ERIKA. BAT
MCR>SUB/PRIO: 150 = ERIKA. BAT
```

Команда SUBMIT ставит пакетное задание с именем ERIKA.BAT в пакетную очередь с именем по умолчанию BATCH с приоритетом 150. Приоритет задания не влияет на приоритет выполнения задачи в пакетном режиме, а влияет на положение задания в очереди.

П р и м е р 2 .

```
DCL> SUBMIT/NAME: MORAN MIK. BAT, ERIKA. BAT
MCR>SUB MORAN=MIK. BAT, ERIKA.BAT
```

Демонстрируется различие между пакетными заданиями пользователя и пакетным заданием QMG.
Пакетное задание QMG получает имя MORAN и состоит из пакетных заданий пользователей MIK. BAT и ERIKA. BAT.

Для передачи команд и данных интерпретаторам командных строк процессор пакетной обработки использует механизм виртуального терминала. Таким образом, команды и данные, содержащиеся в пакетном задании, должны соответствовать полному интерактивному сеансу работы на терминале. Другими словами, пакетное задание должно выполнять:

•регистрацию в системе;
•команды пакетного задания;
•передачу данных задачам;
•обработку кодов состояния, возвращаемых задачами;
•выход из системы.

Команды и данные в пакетном задании могут быть либо командами интерпретаторов командных строк, в частности DCL или MCR, либо специальными командами пакетной обработки.

Термин «данные» включает в себя не только данные, которые необходимо обработать в программе, но и ответы на подсказки, поступающие от команд CLI или от косвенных командных файлов. Следовательно, вся информация, выводимая на печать во время интерактивного сеанса, за исключением команд, называется данными.

Ф о р м а т командной строки пакетного задания:

```
$ [метка:]  [команда]
```

Все команды в пакетных заданиях должны начинаться со знака «$», который информирует процессор пакетной обработки о том, что строка должна интерпретироваться как команда, а не как данные. Знак «$» не является частью команды.

Команды пакетной обработки $JOB и $EOJ выполняют функции регистрации и выхода из системы.

К специальным командам пакетной обработки относятся команды:

```
JOB     CONTINUF            !          EOD    SET
EOJ     GOTO                           STOP   IF
DATA    ON
```

Команда JOB указывает на начало пакетного задания пользователя и должна быть указана первой в каждом файле пакетного задания. Ф о р м а т :

```
$JOB[/[ключ]  [метка]  [[UIC]]
```

где метка — содержит до шести буквенно-цифровых символов. Эта метка используется при регистрации пакетного задания;

UIC — код идентификации, под которым издавалась команда SUBMIT. UIC необходимо заключать в квадратные скобки.

Если UIC не указывается, то любой пользователь, имеющий доступ к файлу в режиме «чтение», может запустить это задание на пакетную обработку.

Для непривилегированного пользователя UIC должен совпадать с UIC, под которым была введена команда SUBMIT.

К л ю ч команды:

```
/TIME: (нн: мм)
```

ИЛИ

```
/TIME :мм
```

где нн:мм, мм — ограничивает время процессора, выделяемого заданию; нн — часы; мм — минуты. По умолчанию — три минуты.

Команда EOJ указывает на логический конец пакетного задания пользователя. Ф о р м а т :

```
$EOJ
```

Если метка конца файла встречается перед командой EOJ, то процессор пакетной обработки воспринимает ее как команду EOJ. Если пакетное задание QMG содержит несколько пакетных заданий пользователя, то процессор пакетной обработки регистрирует выход из системы первого пакетного задания, прежде чем запускается выполнение следующего пакетного задания в цепочке заданий.

Команда DATA указывает на начало блока данных, включенного в пакетное задание, который заканчивается командой EOD.

DATA — необязательная команда. Если она используется в пакетном задании, то должна стоять непосредственно после команды, которая запрашивает входные данные. Ф о р м а т :

```
$DATA /ключ
```

К л ю ч и команды:

```
/NOCOPY
/DOLLARS  [:«строка»]
```

где /NOCOPY — этот ключ запрещает распечатку блоков данных в файле регистрации. По умолчанию — /COPY;
/DOLLARS [:«строка»] — этот ключ разрешает пользователю использовать данные, начинающиеся со знака «$» в первой позиции блока данных. В противном случае процессор пакетной обработки воспринимает первую строку, начинающуюся со знака «$», как конец блока данных.

Ключ /DOLLARS также разрешает использовать аргумент «строка» (длиной от 1 до 15 символов), как ограничитель блока данных.

Процессор пакетной обработки игнорирует команду $EOD до тех пор, пока не будет обнаружена указанная «строка».

Команда EOD обозначает конец блока данных. Ф о р м а т :

```
$EOD
```

Команда STOP останавливает выполнение пакетного задания. Команда STOP может быть использована как отдельно, так и вместе с командами ON или IF. Ф о р м а т ы :

```
$STOP
$ON код состояния THEN STOP
$IF код состояния THEN STOP
```

Если процессор пакетной обработки получает команду STOP, то он воспринимает эту команду как команду, эквивалентную команде $EOJ. В данном пакетном задании может быть несколько команд STOP, если использованы команды GOTO, ON или IF. Если пакетное задание QMG содержит одно пакетное задание пользователя, то процессор пакетной обработки начинает выполнение следующего пакетного задания пользователя.

Команда CONTINUE не вызывает выполнение какого-либо действия. Ее можно использовать как самостоятельно, так и вместе с командами ON или IF. Ф о р м а т ы :

```
$ CONTINUE
$ON WARNING THEN CONTINUE
```

В большинстве случаев данная команда используется вместе с командами IF или ON. По команде CONTINUE процессор пакетной обработки продолжает выполнение задания после получения от задач кода, отличного от SUCCESS.

Команда GOTO позволяет процессору пакетной обработки осуществить переход к строке с заданной меткой и продолжить обработку с этого места. Данная команда может быть использована самостоятельно или вместе с командами ON или IF. Ф о р м а т :

```
$GOTO метка
.
.
.
$ метка:   [команда]
```

где метка — метка, которая включает от одного до шести алфавитно-цифровых символов и завершается двоеточием (:).

Команда ON проверяет код состояния завершения, передаваемый задачами и командами, и указывает на то, какие действия должен выполнять процессор пакетной обработки в этом случае. По умолчанию устанавливается $ON WARNING THEN STOP. Ф о р м а т :

$ON код состояния THEN STOP
CONTINUE
GOTO метка

где код состояния — коды состояния, которые по уровням действия располагаются в следующем порядке:

SUCCESS;
WARNING;
ERROR;
SEVEREERROR.

При указании кода ERROR процессор пакетной обработки игнорирует коды WARNING.

При указании кода SEVEREERROR игнорируются все коды: ERROR и WARNING. Код SEVEREERROR должен указываться в команде как одно слово.

Указание кода SUCCESS в команде не разрешается. Если ни один из кодов состояния не передается, то процессор пакетной обработки подразумевает код SUCCESS. По умолчанию код состояния в команде ON принимает значение WARNING.

Команда SET разрешает или запрещает действие команды ON. Ф о р м а т :

$SET [NO] ON

где SET ON — команда SET ON возвращает процессор пакетных заданий в исходное состояние, установленное ранее по предыдущей команде ON и отмененное по команде SET NO ON;

SET NO ON — команда SET NO ON запрещает действие команды ON, устанавливая действие этой команды по умолчанию. Процессор пакетной обработки игнорирует возвращаемые коды состояния. Допускается указание NO ON (через пробел) или NOON (без пробела).

Команда SET [NO] ON может использоваться с командой оператора и ее действие в пакетном задании то же самое, что и в интерактивном режиме. Несмотря на то что команда SET NO ON запрещает действие команды ON, любое сообщение об ошибке, возвращаемое командами или задачами, будет включено в файл регистрации.

Команда SET NO ON не влияет на команду IF, содержащуюся в задании.

Команда IF проверяет код состояния, который получен при выполнении пакетного задания. В команде IF указываются действия, которые необходимо выполнить процессору пакетной обработки в соответствии с полученным кодом. Ф о р м а т ы :

$IF код состояния THEN STOP
CONTINUE
GOTO метка

где код состояния — коды состояния, имеющие значения:

SUCCESS;
WARNING;
ERROR;
SEVEREERROR.

Использование в команде IF кода состояния SUCCESS не разрешается. Если не возвращается ни один из кодов состояния, то по умолчанию подразумевается код SUCCESS.

Знак комментария. Восклицательный знак (!) используется для включения комментариев в файл регистрации пакетного задания. Это единственный символ, который допускается использовать в пакетном задании пользователя в качестве знака комментария.

Процессор пакетной обработки не передает текст, перед которым стоит восклицательный знак (!), интерпретатору CLI. Комментарии, начинающиеся с любого другого символа, передаются интерпретатору CLI и могут вызвать ошибки в работе процессора пакетной обработки. Ф о р м а т ы :

$! комментарий
$ [метка:] команда задания! комментарий

Поле комментария может содержать любые символы. Процессор пакетной обработки не обрабатывает информацию, стоящую после восклицательного знака (!).

Косвенные командные файлы. Символ @ в пакетном задании обрабатывается как признак косвенного командного файла. Когда выполняется командная строка, косвенный командный файл становится допустимым и выполняются команды, расположенные в нем. Ф о р м а т :

$@имя файла [.CMD]

П р и м е р :

$JOB

```
$!
$! Пример задания
$!
$ON ERROR THEN STOP
$RUN ATASK
$DATA
ALAN
JANE
JIM
TOM
$EOD
$EOJ
```

Запускается задача ATASK, в которую последовательно вводятся текстовые данные. В случае возникновения ошибки пакетное задание завершается.

6.10.3. ДОПОЛНИТЕЛЬНЫЕ КОМАНДЫ QMG

Команда **SHOW QUEUE(QUE/LI)** распечатывает на терминале информацию в заданиях на печать и пакетных заданиях. Ф о р м а т ы :

```
DCL>SHOW QUEUE [имя очереди] /ключ(и)
MCR>QUE  [имя очереди:] '[[UIC]]   [имя задания] /ключ(и)
```

К л ю ч и DCL и MCR:

Ключи DCL	**Ключи MCR**
/FULL	/FU
/FILES	/LI
/BRIEF	/BR
/DEVICE	/LI:P
/ENTRY: NNN	/EN: NNN
/FORM [:N]	/FOC: N
/NAME: Имя задания	имя задания
/OWNER_UIC:UIC	[UIC]
/PRINT	/LI:P
/BATCH	/LI: B

Команда **SHOW PROCESSOR (QUE/LI:DEV)** выводит на терминал информацию о характеристиках процесса пакетной обработки, об устройстве печати, а также о других устройствах, работающих под управлением QMG. Ф о р м а т ы :

```
DCL>SHOW PROCESSOR/ключ
MCR>QUB [имя процессора:] /ключ
```

К л ю ч и DCL и MCR:

Ключи DCL	**Ключи MCR**
Имя процессора [:]	/LI: DEV
/BATCH	/LI: DEV: B
/PRINT или /DEVICE	/LI: DEV: P
/INPUT	/LI: DEV: I

Команда SET QUEUE (QUE/MOD) позволяет модифицировать атрибуты заданий на печать или пакетных заданий, либо файлов, содержащих задания в очередях. Такие задания и файлы устанавливаются в очередь командами PRINT и SUBMIT. Ф о р м а т ы :

```
DCL>SET QUEUE имя очереди: имя задания/ключ  [/ключ(и)]
DCL>SET QUEUE /ENTRY: NNN/ ключ  [/ключ(и)]
MCR>QUE имя очереди: имя задания/MOD/ключ [/ключ(и)]
MCR>QUE /EN: NNN/MOD/ ключ [/ключ(и)]
```

К л ю ч и DCL и MCR:

Ключи (DCL)	**Ключи MCR**
/AFTER: (чч:мм дд-ммм-гг) .	/AF:чч:мм:дд-ммм-гг
/JOBCOUNT: N	/CO: N
/FORMS: N	/FO: N
/LENGTH: N	/LE: N
/LOWERCASE	/LO

/PAGE_COUNT: N	/PA: N
/PRIORITY: N	/PRIO: N
/[NO] RESTART	/[NO] RE
/UPPERCASE	/NOLO

Ф о р м а т ы команды для файлов:

DCL>SET QUEUE/ENTRY: NNN/FILE_POSITION: N /ключ [/ключ(и)]
DCL>SET QUEUE имя очереди имя: задания/FILE POSITION: N
/ключ— [/ключ (и)]
MCR>QUE /EN:NNN/MOD/FI:N /ключ [/ключ(и)]
MCR>QUE имя очереди: имя задания /MOD/FI: N /ключ [/ключ(и)]

К л ю ч и DCL и MCR:

Ключи DCL	**Ключи MCR**
/COPIES: N	/CO: N
/[NO] DELETE	/[NO] DEL

При вводе команд PRINT или SUBMIT пользователь указывает атрибуты задания QMG. Команда SET QUEUE изменяет соответствующие атрибуты неактивного задания.

Указав ключ /FILE_POSITION: N(/FI : N) в командной строке SET QUEUE, пользователь может изменять атрибуты файлов, составляющих задание на печать.

Команда HOLD (QUE/HO) задерживает задание в очереди до тех, пока оно не будет выведено из этого состояния. Ф о р м а т ы :

DCL>HOLD/имя очереди имя задания
DCL>HOLD/ENTRY: NNN
MCR>QUE имя очереди:имя задания
MCR>QUE/EN: NNN/HO

Команда RELEASE (QUE/RE) освобождает задания, находящиеся в очереди в задержанном состоянии. Ф о р м а т ы :

DCL>RELEASE/JOB имя очереди: имя задания
DCL>RELEASE/ENTRI: NNN
MCR>QUE имя очереди:имя задания
MCR>QUE/EN: NNN/REL

Команда DELETE (QUE/DEI) удаляет файлы или пакетные задания, указанные именами или идентификаторами (номерами записей заданий), из очереди. Ф о р м а т ы :

DCL>DELETE/имя очереди: имя задания [/FILE_POSITION: N]
DCL>DELETE/ENTRY: NNN [/FILE_POSITION: N]
MCR>QUE имя очереди: имя задания /FI: N/DEL
MCR>QUE /EN: NNN/FI: N/DEL

Данные команды удаляют задание из очереди путем указания идентификатора задания или имени задания. Кроме того, указав ключ /FILE_POSITION: N (/FI: N), можно удалить отдельный файл в задании.

Непривилегированные пользователи могут удалить только свои собственные задания, в то время как привилегированные пользователи — любое задание. В одной очереди может быть указано несколько заданий с одним и тем же именем.

Команда DELETE удаляет из очереди первое задание с указанным именем. Задания, имеющие одно и то же имя, различаются по их идентификаторам.

7. Системные обслуживающие программы

7.1. ОБЩИЕ СВЕДЕНИЯ ОБ ОБСЛУЖИВАЮЩИХ ПРОГРАММАХ ОСРВМ

Для широкого круга пользователей СМ ЭВМ качество операционной системы во многом определяется ее сервисными средствами. В ОСРВМ такими средствами являются удобные и достаточно простые обслуживающие программы (утилиты), рассчитанные на программистов и операторов различной квалификации. С помощью этих программ создают, преобразовывают и редактируют файлы, переписывают их с носителя на носитель, сохраняют и восстанавливают тома, отлаживают программы, обрабатывают библиотеки и т. д. Обслуживающие программы ОСРВМ условно можно разделить на следующие классы:

- •программы обработки файлов;
- •программы проверки и подготовки носителей;
- •программы сохранения и восстановления томов;
- •отладочные программы;
- •библиотекарь;
- •программы редактирования.

Программы редактирования служат для создания и корректировки символьных файлов. В состав ОСРВМ входит несколько программ-редакторов: текстовый, экранный, универсальный, программируемый и пакетный. Работа с каждым из них достаточно проста, однако описание их возможностей и набора команд так велико, что выходит за пределы, допускаемые объемом данного справочника.

Работе с различными обслуживающими программами присущи две общие черты: формат командной строки и способы запуска обслуживающих программ.

Ф о р м а т командных строк для обслуживающих программ:

выв. ф1 [, ..., выв. ф N] =вв. ф1 [, ..., вв. ф N]

В зависимости от применяемой программы формат командной строки может быть изменен. Так, большинство обслуживающих программ используют косвенные командные файлы, а некоторые программы — сокращенную форму командной строки.

Число спецификаций файлов в командных строках зависит от конкретной программы, но общая длина строки ограничена. Каждая спецификация файла имеет стандартный формат, принятый в ОСРВМ:

устр: [гр, чл] имя файла, тип; версия/ключ, ..., /подключ

Подробно все компоненты спецификации файла разбираются в разд. 1 и 3.

Как правило, функции обслуживающих программ задаются с помощью ключа и подключей. Знак «— » и символы NO отвергают действие ключа. Ключи могут иметь аргументы в символьном и цифровом виде. Цифровые значения аргументов по умолчанию восьмеричные. Для указания десятичного числа его необходимо ограничить десятичной точкой. Подключ — двухсимвольное имя — уточняет действие ключа; синтаксически правила ввода ключей и подключей аналогичны.

П р и м е р :

PIP>[200, 200] *.*;*/PR/FO

где подключ /FO используется совместно с ключом /PR.

Обслуживающие программы в ОСРВМ начинают работу по командам оператора или через косвенный командный файл. Обслуживающие программы запускаются в ответ на подсказку программы связи с оператором (MCR). Установленные программы запускаются двумя способами:

>имя программы командная строка

или

>имя программы

где > — подсказка MCR.

При первом способе запуска программа выполняет требуемые действия и завершается, а при втором — командная строка может вводиться многократно до тех пор, пока оператор не завершит обслуживающую программу управляющим символом CTRL/Z. Неустановленные обслуживающие

программы запускаются с помощью команды MCR RUN. Ф о р м а т :

RUN $ имя программы

Программа после получения управления печатает подсказку

имя обслуживающей программы>

и ждет ввода командной строки.

Если обслуживающей программе нужно работать с кодом UIC, отличным от текущего, то в команде RUN указывается ключ /UIC = [гр, чл]. Так же как и при втором способе запуска установленных программ, командная строка может вводиться оператором многократно. Для большинства обслуживающих программ командные строки могут вводиться и из косвенных командных файлов.

7.2. ПРОГРАММЫ ОБРАБОТКИ ФАЙЛОВ

Программы обработки файлов служат для переписи, преобразования, удаления, сравнения файлов, а также для выполнения других вспомогательных функций, необходимых пользователю при работе с файлами.

7.2.1. ПРОГРАММА PIP

Программа работы с файлами PIP служит для файловой переписи файлов с одного устройства с файловой структурой ОСРВМ на другое, а также для копирования, удаления, переименования, разблокирования файлов и распечатки каталогов. Формат командной строки PIP может меняться в зависимости от выполняемых функций. Несколько команд, заданных в строке, отделяются друг от друга знаком &.

Умолчания для спецификаций файлов PIP соответствуют общепринятым, но у программы PIP существует ключ /DF, явно определяющий умолчания. При употреблении нескольких файлов в командной строке умолчания в спецификациях последующих файлов соответствуют спецификациям в предыдущих.

Ключи команды глобальные, они могут быть указаны один раз для всего списка файлов с любой стороны от знака равенства.

Употребление специальных символов. Программа PIP разрешает применение символа * в одном или нескольких полях спецификации файла. Символ * используется вместо явного задания поля и означает соответственно все файлы, все типы и т. д. Знак % может указываться в поле имени и типа файла. В этих полях знак % означает, что на его месте может быть любой допустимый символ.

Символ * не может использоваться в спецификациях вводных файлов в следующих случаях:

- при конкатенации файлов с указанным файлом;
- в дополнение к существующим файлам;
- при корректировке существующего файла;
- при распечатке каталога.

Ключи программы работы с файлами. Отсутствие в командной строке ключей и подключей означает копирование файла на то же самое или другое устройство или слияние нескольких вводных файлов в выводной файл. Ниже перечисляются ключи программы PIP и дается краткая характеристика их функционирования.

/AP — открывает существующий файл и добавляет файлы ввода в конец существующего файла. Ф о р м а т :

выв.ф = вв.ф1[, вв.ф2,..., вв.фN]/AP/FO

где /FO — подключ принадлежности файла, который указывает, что код UIC владельца выводного файла должен соответствовать коду выводного каталога. Если подключ /FO не указан, то UIC нового файла будет тот, под которым работает программа PIP.

/BS — служит для определения размера блока магнитной ленты. Ф о р м а т :

выв. ф/BS :N [.] = вв. ф

где N — номер блока в байтах.

/CD — используется для копирования файлов, при этом дата создания нового файла берется от старого. Ф о р м а т :

выв. ф/СВ = вв. ф1 [, вв. ф2, ..., вв. фN]

/DD — осуществляется поиск файлов, которые были созданы в указанный период времени между начальной и конечной датой. Ф о р м а т :

/DD: начальная-дата: конечная-дата

Вместо начальной или конечной даты можно употреблять символ *.

Пример написания даты: 05-JUN-84 (5 июня 1984 года).

/DE — служит для удаления файлов. Задав подключ /LD, можно распечатать на терминале имена удаляемых файлов. При удалении файлов умолчание в поле версии недопустимо. Ф о р м а т :

вв. ф1 [, вв.ф2, ..., вв. фN]/DE [/LD]

/DF — позволяет установить умолчание для устройства или кода UIC в спецификациях файлов. Ф о р м а т :

устр.: [UIC] /DF, или устр: /DF, или [UIC] /DF, или /DF

В последнем случае по умолчанию устанавливается устройство SYO: и UIC, под которым работает программа PIP.

/EN — обеспечивает ввод синонима для файла, разрешая, таким образом обращение к файлу с помощью нескольких имен. Ф о р м а т :

выв.ф = вв.ф1 [, вв.ф2,..., вв.фN]/EN [/NV]

где /NV — подключ новой версии.

В спецификации выводного файла имя, тип и версия могут задаваться явно, символом * или быть пустыми. Наличие умолчания или символа * означает, что должны использоваться соответствующие поля вводного файла. Для выводного и вводных файлов нельзя указывать разные устройства.

/EOF — позволяет записать признак конца файла на указанное место, в результате сохраняется часть разрушенного файла. Ф о р м а т :

вв.ф/EOF [:блок: байт] [, ..., вв.ф/EOF[:блок : байт]]

где блок — номер блока, в котором устанавливается признак конца файла;— байт — номер байта (не более 512), в котором определяется признак конца файла. Если номера блока и байта не указаны, то признак конца файла устанавливается в последний байт последнего блока непрерывного файла.

/EX — удаляет из процесса поиска указанные файлы. Из поиска можно исключить Целые классы имен, типов и версий, указывая в соответствующих полях символ *. В спецификации нельзя указывать*. *; *, нельзя также указывать только устройство и каталоги. Заданный ключ /EX действует далее по умолчанию. Ф о р м а т :

спецификация файла /EX

/FI — разрешает доступ к файлу по его идентификатору. Ф о р м а т :

выв. ф=/FI: ном. ф: посл. ном. ф

где ном. ф — номер файла;
посл. ном. ф — последовательный номер файла.

/FR — распечатывает общее число свободных блоков на устройстве, наибольший размер файла и количество занятых заголовков файла. Ф о р м а т :

устр: /FR

/ID — выдает на терминал номер версии программы PIP и сообщает о возможности работы с магнитной лентой.

/LI, /BR, /FU, /TB — ключи распечатки каталогов. Выводят один или более каталогов в различных форматах. Ф о р м а т :

[файл листинга] = [вв. ф1] [, вв. ф2,..., вв. фN] /ключ [/подключ]

Умолчание для файла листинга — TI:. умолчание для вводного файла *. *; *. По ключу /BR дается краткая распечатка каталога — имя файла, тип, версия. По ключу /LI в дополнение к этой информации распечатывается:

число использованных блоков (десятичное);

код файла: пробел — прерывный, С — непрерывный, L — блокированный;

дата и время создания файла;

завершающая строка, которая содержит число использованных блоков, число зарезервированных блоков и общее число файлов в каталоге (все числа десятичные).

/FU [:N] — по этому ключу распечатывается каталог в полном формате, N — указывает число символов в строке. В дополнение к информации, печатаемой по ключу /LI, выводится:

идентификатор файла в формате — номер файла, последовательный номер файла;

код идентификации владельца файла ([гр, чл]);

код защиты файла в формате «системная», «владелец», «групповая», «общая» (см. ниже ключ /PR). Каждое поле защиты содержит комбинацию значений:

R, W, E, D,

где R — чтение разрешено; W — запись разрешена; E — расширение разрешено; D — удаление разрешено;

дата и время создания файла;

дата и время последней корректировки и число корректировок.

/NM — позволяет не печатать некоторые сообщения об ошибках. Ф о р м а т :

вв.1 [, вв.2, ..., вв. фN] [/ключ] /NM

где ключ — некоторые комбинации ключей и подключен (/LI, /DE, /PU и т. д.).

/PR — позволяет изменять привилегии доступа к файлу. Защита обеспечивается для четырех категорий пользователей: «системная», «владельца», «групповая», «общая». Для каждой категории может быть указано право на чтение, запись, расширение и удаление файла. Ф о р м а т :

вв.ф/PR [/SY[:RWED]] [/OW[:RWED]] [/GR[:RWEDj] [/WO[:RWED]] [/FO]

где вв. ф — спецификация вводного файла должна быть задана явно;
/SY — подключ системной категории;
/OW — подключ собственной категории;
/GR — подключ групповой категории;
/WO — подключ общей категории.

С помощью этих подключен пользователь указывает какая из категорий защиты должна быть изменена:

значения RWED имеют тот же смысл, что и в ключе /FU и указывают, какие права допустимы для соответствующей категории пользователей.

/FO — подключ принадлежности файла владельцу. Он устанавливает код идентификации владельца файла, равный коду UIC каталога, в котором файл указан.

/PU — обеспечивает возможность удаления устаревших версий файла. Ф о р м а т :

вв.ф1 [, вв.ф2, ..., вв.фN] /PU : N [/LD]

Если указано значение N и последняя версия файла имеет номер M, то версия с номерами, меньшими или равными M — N, удаляются. Значение N применимо к спецификации файла, непосредственно предшествующей ему (локально). При задании подключа /LD имена удаляемых файлов будут распечатываться на терминале.

/RE — обеспечивает возможность изменения имени и типа файла. Ф о р м а т :

выв.ф = вв.ф1 [, вв.ф2, ..., вв.фN] /RE [/NV]

С помощью подключа /NV пользователь может увеличить на один номер версию переименованного файла по сравнению с последней версией существовавшего файла. Переименование файлов должно производиться на одном устройстве.

/RM — позволяет убирать запись из каталога. В отличие от /DE этот ключ не стирает файл, а удаляет только запись из каталога. Ф о р м а т :

вв.ф1, [, вв.ф2, ..., вв.фN] /RM

/RW — служит для перемотки магнитной ленты. Ф о р м а т :

выв.ф [/RW]=вв.ф (/RW]

В том случае, когда ключ /RW задач со спецификацией выводного файла, лента перематывается на начало, когда же ключ задач со спецификацией вводного файла, лента позиционируется перед этим файлом.

/SB — при переписи файла с магнитной ленты на диск позволяет пересекать записям границы блоков. Ф о р м а т :

выв.ф/SB = вв.ф

/SD — в процессе диалога с оператором удаляет или оставляет файлы в каталоге. Ф о р м а т :

вв.ф1 [, вв.ф2, ..., вв.фN] /SD

Программа PIP для каждого удаляемого файла запрашивает подтверждение:

DELETE FILE вв.ф [Y/N/G/Q]

Оператор должен ответить, нажав клавишу с одной из перечисленных ниже букв:
Y — удалить файл и продолжить просмотр;
N — сохранить файл и продолжить просмотр;
G — удалить все версии файла и завершить просмотр;
Q — сохранить файл и завершить просмотр.

Если в комбинации с этими буквами оператор набирает символы CTRL/Z, то после выполнения заданной функции происходит выход из программы PIP.

/SP — задает список файлов, выводимых на печать асинхронно с помощью задачи системного вывода. Ф о р м а т :

выв.ф = вв.ф1 [, вв.ф2,..., вв.фN] /SP [:M]

где выв.ф — спецификация файла, выводимого на печать; M — число копий вывода.

/SR — позволяет прочитать файл, который был открыт для записи другой задачи. Ф о р м а т :

выв.ф = вв.ф/SR

/TD — осуществляет поиск только тех файлов, которые были созданы за текущий день. Ф о р м а т :

/TD

/TR — позволяет усекать конец файла до логического признака конца. Ф о р м а т :

вв.ф1 [, вв.ф2,..., вв.фN] /TR

/UF — используется для создания каталога файлов на томе. При включении файла в каталог должна быть указана принадлежность файла (подключ /FO). Ф о р м а т :

выв.ф. [/UF [/FO]]=вв.ф1, вв.ф2,..., вв.фN

/UN — позволяет разблокировать файл, который был заблокирован в результате некорректного закрытия. Ф о р м а т :

вв.ф1 [, вв.ф2,..., вв.фN] /UN

/UP — позволяет записывать в существующий файл новые данные от его начала. Ф о р м а т :

выв.ф = вв.ф1 [, вв.ф2,..., вв.фN] /UP[/FO]

где /FO — подключ принадлежности файла.

7.2.2. ПРОГРАММА ПРЕОБРАЗОВАНИЯ ФАЙЛОВ FLX

Программа FLX производит файловые преобразования из форматов систем ДОС СМ и РАФОС в форматы файловой структуры ОСРВМ, и наоборот:
из ДОС КП с ОСРВМ и, наоборот, из ОСРВМ в ДОС КП;
из РАФОС (ФОБОС) в ОСРВМ и, наоборот, из ОСРВМ в РАФОС.

Кроме того, программа преобразования файлов выполняет дополнительные функции:
•распечатывает каталоги томов и кассет РАФОС и ДОС СМ;
•удаляет файлы из томов ДОС СМ и РАФОС;
•инициирует тома ДОС СМ и РАФОС.

Программа FLX требует не менее 8 Кслов.

Командная строка FLX состоит из необязательной спецификации выводного файла и нескольких спецификаций вводного файла в следующем формате:

выв.ф/ключ = вв.ф1/ключ [, вв.ф2/ключ,..., вв.N/ключ]

Допустимый формат спецификаций выводного файла:

устр: [гр.чл]

Символ * употребляется только в вводных файлах, номера версий используются только для файлов

ОСРВМ и не могут заменяться символом *.

Ниже приводятся ключи программы преобразования файлов. Программа FLX работает с тремя типами ключей для файловых передач.

1. Ключи типа формата определяют формат файла: /DO — идентифицирует файл в формате ДОС СМ, /RS — в формате ОСРВМ, /RT — в формате РАФОС.

По умолчанию предполагается ключ /DO для вводных файлов, /RS — для выводных файлов. Чтобы изменить умолчания на противоположные, необходимо задать ключ FLX>/RS.

Пример:

```
FLX>DP1 : =DK3 : FILE1 .MAC
Файл FILE 1.MAC переписывается с тома ДОС СМ (DK3) на том
ОСРВМ (DP1:).
```

2. Ключи типа передач управляют форматом передачи: двоичным, символьным, форматом образа.

/FA:N — форматный символьный ключ. Символьные файлы ДОС и РАФОС должны иметь форматный тип. Он определяет записи в семибитном символьном коде. Записи заканчиваются символами возврата каретки и перевода строки (CR—LF), перевода формата (FF) или вертикальной табуляции (VT). В передачах файлов на ДОС или РАФОС в ОСРВМ пары CR-LF удаляются в начале и конце записей. В передачах из ОСРВМ в ДОС РАФОС пары CR—LF добавляются в конец записей, не имеющих в конце символов LF или FF. N — размер записи. Если N не указан для выводного файла ОСРВМ, то генерируются записи переменной длины.

/FB:N — форматный двоичный ключ. Двоичные файлы ДОС и РАФОС должны быть форматными. В этом случае форматные двоичные заголовки и контрольные суммы добавляются при выводе записей в файлы ДОС или РАФОС и, наоборот, они убираются при передаче файлов в ОСРВМ.

/IM:N — ключ формата образа. При передаче данных в формате образа создаются непрерывные файлы с записями фиксированной длины (по умолчанию N = 512 байт).

Подразумеваются также типы файлов по умолчанию:

Ключ	**Тип файлов**
/IM	.TSK, .OLB, .MLB. .SYS, .SML, .ULB, .EXE
/FB	.OBJ, .STB, .BIN, .LDA
/FA	Все другие

3. Ключи управления файлом служат для управления обработкой файлов.

/BL:N — указывает число смежных блоков, которые должны быть выделены для размещения выводного файла; N — число блоков.

/CO — указывает, что выводной файл должен быть непрерывным (имеет смысл только для дисков).

/DE — в комбинации с ключами /DO и /RT используется для удаления файлов с томов ДОС СМ или РАФОС.

/DI (/LI) — в комбинации с ключами /DO или /RT вызывает распечатку каталогов тома ДОС СМ или РАФОС.

/DNS:N — определяет плотность записи на магнитной ленте типа ММ:. При плотности 32 бит/мм необходимо указать N = 800, а при плотности 64 бит/мм— 1600.

/FO — указывает, что должны быть использованы соглашения относительно управления кареткой, принятые в языке Фортран.

/ID — выдает номер текущей версии программы FLX.

/NU:N — указывает число блоков каталога N при инициализации диска РАФОС. Используется вместе с ключами (/RT и /ZE).

/RW — служит для перемотки магнитной ленты на начало.

/SP — указывает, что выводной файл должен обрабатываться задачей системного вывода (только для ОСРВМ).

/UI — указывает, что выводной файл должен иметь тот же UIC, что и вводной.

/VE — предназначен для чтения и проверки каждой записи, выводимой на кассетную магнитную ленту.

/BS:N — указывает размер блока при выводе на кассетную магнитную ленту (по умолчанию 128 байт).

/ZE [:N] — инициирует дома ДОС или РАФОС; N — указывает количество блоков каталога.

7.2.3. ПРОГРАММА СРАВНЕНИЯ ФАЙЛОВ CMP

Программа сравнения файлов построчно сравнивает в символьном коде два файла для того, чтобы определить, являются ли параллельные записи идентичными.

Программа сравнения файлов выводит:

•листинг, в котором указаны различия между файлами. Слева печатаются строки первого файла, у которых есть различия, справа — строки второго файла;

•листинг файла с различиями, отмеченными восклицательными знаками (!);

•командные строки для программы SLP (пакетный редактор). Эти строки содержат команды SLP, с помощью которых можно сделать первый файл идентичным второму.

Ф о р м а т командной строки CMP:

выв.ф/ключ = вв.ф1, вв.ф2

где выв.ф — файл листинга;
вв.ф1 и вв.ф2 — сравниваемые файлы.

Ключи:

/BL — указывает, что пустые строки в обоих вводных файлах включаются в процесс сравнения. По умолчанию /—BL.

/CB — определяет, что программа CMP распечатывает вв.ф2, в котором каждая строка, не идентичная строке вв.ф1, отмечается знаком !. По умолчанию /—CB.

/CO — указывает, что комментарии включаются в процесс сравнения.

/DI — вызывает распечатки всех имеющихся различий в двух файлах.

/FF — указывает, что записи, состоящие из символа FF (перевод формата), включаются в процесс сравнения.

/LI:N — определяет число строк (N) в обоих файлах, которые должны совпадать, чтобы программа CMP опознала конец несовпадения (по умолчанию 3).

/LN — показывает, что строкам в выводном файле предшествуют номера.

/MB — указывает, что все пробелы и знаки табуляции сравниваются.

/SL — по этому ключу программа CMP создает выводной файл для программы SLP.

/SP [:N] — напоминает, что выводной файл будет выводиться программой системного вывода (N — количество копий файла).

/TB — указывает, что все пробелы в конце строки сравниваются.

/VB:NNN — определяет восьмеричный символьный код, используемый как признак несовпадения.

7.2.4. ПРОГРАММА РАСПЕЧАТКИ ФАЙЛОВ DMP

Программа распечатки файлов DMP выводит содержимое томов и файлов на устройство вывода в различных форматах. Программа DMP работает в файловом режиме или в режиме устройства.

При распечатке тома может быть распределена область распечатываемых логических блоков.

В файловом режиме определяется один вводной файл, который распечатывается весь или частично по указанию области виртуальных блоков. При этом вводный том должен быть смонтирован. Данные распечатываются записями или блоками.

В режиме устройства указывается немонтированное устройство вывода и распечатывается вся область логических блоков, заданная ключом /BL : N : M.

Запуск программы DMP производится так, как описано в разд. 7.1.

Ф о р м а т к о м а н д н о й с т р о к и D M P :

[выв.ф] [/ключ] [/ключ...] = вв.ф [/ключ] [/ключ...]

Если спецификация выводного файла отсутствует, создается файл DMPFIL.DMP.

К л ю ч и :

/AS — указывает, что данные будут распечатаны в символьном виде.

/BA: N: M — позволяет определять двухсловный базовый адрес блока, где N и M восьмеричные. Этот ключ используют для блоков, номера которых превышают 16 бит. При этом номера блоков, заданные ключом /BL, будут прибавляться к базовому адресу для получения действительного значения адреса блока. Значение 1 для базового блока соответствует 200000 (8).

/BL: N: M — указывает область выводимых блоков (N— первый, M — последний).

/BY — указывает, что данные вводятся в байтном восьмеричном формате.

/DC — указывает, что данные выводятся в формате десятичных слов.

/DENS: N — определяет плотность записи на магнитной ленте (MM:). При N = 800 плотность — 32 бит/мм, а при N = 1600 — 64 бит/мм.

/FI: нф: пф — используют номер файла (нф) и последовательный номер файла (пф) как идентификатор вводного файла, используемый вместо имени.

/HD[: F: U] — употребляется в файловом режиме для распечатки заголовка файла.

При указании :F (по умолчанию) заголовок распечатывается в формате файловой структуры OCPBM, при указании :U заголовок распечатывается в неформатированном восьмеричном виде.

/HF — форматирует только блоки данных, являющиеся заголовками файлов OCPBM.

/HX — указывает, что данные выводятся в шестнадцатеричном байтовом формате.

/ID — идентифицирует версию программы DMP.

/LB — выводит номер логического блока, с которого начинается файл.

/LC — распечатывает данные прописными буквами английского алфавита.

/LW — выводит данные в формате двойных шестнадцатеричных слов.

/MD [:N] — управляет при распечатке номерами строк. Этот ключ позволяет не сбрасывать в нуль значение номера строк на границе блока; N задает значение первой строки (по умолчанию 0).

/OCT — распечатывает данные в восьмеричном формате, не подавляя другие заданные форматы, кроме /AS.

/R5 — выводит данные в формате RADIX = 50.

/RC — указывает, что данные выводятся записями, а не поблочно.

/RW— перематывает ленту к началу перед обращением к ней.

/SB: N, где N — указывает число блоков, которое нужно пропустить вперед (N) или назад (—N) на ленте.

/SF: N, где N — указывает число признаков EOF, которое нужно отсчитать при перемотке ленты вперед (N) или назад (-N).

/SP — указывает, что файл выводится на печать с помощью задачи системного вывода.

7.3. ПРОГРАММЫ ПРОВЕРКИ И ПОДГОТОВКИ НОСИТЕЛЕЙ

7.3.1. ПРОГРАММА ФОРМАТИРОВАНИЯ ДИСКА FMT

Программы служат для форматирования и проверки дисковых томов типа DK:, DM:, DP:, DY:, DR:.

Программа выполняет следующие функции:

•пишет полный заголовок для каждого сектора на диске при форматировании;

•проверяет содержимое адреса заголовков каждого сектора;

•устанавливает плотность записи на диске DY;

•задает максимально допустимое число ошибок для диска;

•запускает программу BAD через директивы порождения задач.

Запуск программы FMT производится по команде MCR. Ф о р м а т :

FMT DDN: [/ключ1.../ключN]

где DDN: — мнемоника и номер форматируемого тома.

К л ю ч и :

/BAD — вызывает задачу обнаружения дефектных блоков.

/DENS=[HIGH, LOW] — устанавливает высокую или низкую плотность записи на дискете (DY:), по умолчанию — низкая плотность (LOW).

/ERL — задает предельное число ошибок для форматируемого тома; при достижении этого числа программа FMT выдает сообщение и заканчивает работу.

/MAN — служит для ручного режима работы FMT и позволяет форматировать отдельный сектор или дорожку диска.

/NOVERIFY— отменяет операции по проверке чтения.

/OVR — заставляет игнорировать блок описателя дефектов (последний), записанный при изготовлении диска (см. разд. 7.3.2).

/VE — запускает операции чтения для проверки корректности записей заголовков.

/WLT: N — перезаписывает блок описателя дефектов на диске DM: после того, как программа FMT добавит обнаруженные дефектные секторы к имеющимся в блоке описателя дефектов. Десятичное число N используется как порядковый номер тома.

/@Y —требует ввода команд для программы FMT из косвенного командного файла.

7.3.2. ПРОГРАММА ПОИСКА ДЕФЕКТНЫХ БЛОКОВ BAD

Программа BAD проверяет дисковые пакеты для определения числа и места дефектных блоков на носителе. Эта информация записывается в пригодный для использования блок на диске и при инициации тома дефектные блоки отмечаются как занятые. Команда MCR INI распределяет их в файл [0,0] BADBLK.SYS. Ф о р м а т :

BAD DDUU: [/ключ...]

где DD — имя устройства;
UU — номер устройства.

Все ключи для устройства должны быть заданы в одной строке. Перед запуском программы BAD диск должен быть сформатирован.

К л ю ч и :

/ALO: метка тома — запрашивает ввод дополнительных дефектных блоков. При использовании этого ключа нет необходимости в повторной инициализации диска, работа ведется с монтированным устройством и пользователь должен быть привилегированным.

/LI — распечатывает на терминал номера всех дефектных блоков, которые были обнаружены во время проверки устройства.

/MAN — запрашивает ввод информации о дефектных блоках, выдавая подсказку на терминал, а затем, проверяя диск, включает в блок описателей дефектных блоков информацию, введенную с терминала.

/OVR — игнорирует информацию на последней дорожке. Описатель дефектных блоков записывается в последний пригодный для использования блок перед последней дорожкой, т. е. диск DM (с последней дорожкой) рассматривается как устройство, не имеющее ее.

/PAT = M:N — служит для поиска дефектных блоков с помощью двухсловных тестовых данных, вводимых пользователем как числа M и N (шестнадцатибитные восьмеричные). Стандартно используются тестовые данные 165555 и 133333.

/RETRY — предназначен для корректировки ошибок оборудования средствами драйвера. Следовательно, устранимые ошибки, такие, как ошибки контрольной суммы, могут быть устранены и блок будет отмечен как годный для использования.

/UPD — служит для считывания блока описателя дефектных блоков и выдачи подсказки на ввод дополнительных дефектных блоков. При этом BAD не проверяет диск.

Описываемые ниже три ключа работают только с версией программы BAD, ориентированной на работу без операционной системы:

/CSR= NNNNNN, где NNNNNN — новый восьмеричный адрес регистра команд и состояния устройства;

/VEC = NNN, где NNN — новый восьмеричный адрес вектора прерываний устройства. Эти ключи используются при нестандартном подключении устройства;

/WCHK — вызывает контроль записи для каждой записи на устройстве.

7.3.3. ПРОГРАММА ПРОВЕРКИ ФАЙЛОВОЙ СТРУКТУРЫ VFY

Программа проверки файловой структуры позволяет: «установить возможность чтения тома и корректность файловой структуры на нем;

•напечатать число свободных блоков на томе;

•найти файлы, которые числятся в индексном файле, но не указаны ни в одном из каталогов;

•распечатать все файлы с указанием идентификатора, имени и кода UIC владельца;

•отметить как использованные блоки, которые не заняты файлом, но распределены для него;

•перестроить файл распределения пространства диска так, чтобы адекватно отразить информацию в индексном файле;

•сбросить все признаки удаления файла;

•выполнить проверку чтения каждого блока на томе;

•проверить корректность каталогов;

•проверить дефектные заголовки файлов.

Во время работы программы VFY на смонтированном томе не должны выполняться никакие другие операции. Ф о р м а т :

файл листинга, устр. для временного хранения файла = проверяемый том/ключ или проверяемый том/ключ

Временный файл используется программой VFY во время проверки и поиска потерянных файлов. По умолчанию создается на устройстве SY0: и после завершения программы стирается.

К л ю ч и программы VFY:

Если в команде ключ не указан, то проверяется возможность считывания и корректность файловой структуры тома. При этом считываются все заголовки файла в индексном файле и проверяется, какие дисковые блоки, указанные в области планирования в заголовке файла, отмечены как распределенные.

/DE — позволяет сбросить индикатор удаления в заголовке файла для тех файлов, которые отмечены для удаления, но не удалены. При этом том монтируется с ключом /UNL, и программа VFY выполняется под системным кодом идентификации.

/UP — позволяет отметить, как занятые под файлы блоки, которые числятся свободными, но в действительности распределены для файла.

Примечания: 1. Файлы с многократно распределенными блоками должны быть удалены до корректировки. 2. Временный файл должен находиться на другом томе. 3. Программа VFY должна выполняться под системным кодом идентификации. 4. Корректируемый том должен быть доступен для записи.

/RE — позволяет восстановить потерянные блоки, которые отмечены как занятые, но никакие файлы их не содержат.

/FR — распечатывает размер доступного свободного пространства на томе.

/LO — дает возможность просмотреть весь том, чтобы отыскать те файлы, которых нет ни в одном каталоге, т. е. потерянные в том смысле, что к ним нельзя обратиться по имени. Если в томе существует каталог [1, 3], то эти файлы будут введены в него.

/LI — распечатывает из индексного файла идентификаторы файлов (номер файла, последовательный номер файла, имя и UIC владельца).

/RC [:N] — проверяет, может ли каждый блок каждого файла на томе быть прочитан. :N — блочный фактор, указывающий число блоков, которые должны быть считаны за один раз.

/DV — проверяет каждый каталог на томе и сообщает о несуществующих файлах, перечисленных в каталоге.

/HD [AL]—определяет дефектные заголовки файлов на томе. Если задать подключ /AL, все дефектные заголовки файлов будут удалены.

7.4. ПРОГРАММЫ СОХРАНЕНИЯ И ВОССТАНОВЛЕНИЯ ТОМОВ

7.4.1. ПРОГРАММА КОПИРОВАНИЯ И ВОССТАНОВЛЕНИЯ ТОМОВ (BRU)

Системная программа BRU позволяет копировать и восстанавливать тома с файловой структурой ОСРВМ. Операции копирования и восстановления допустимы для томов, расположенных на магнитных дисках и магнитных лентах:

•с диска на ленту — операции копирования;
•с ленты на диск — операции восстановления;
•с диска на диск — операции копирования и восстановления.

Наряду с этими основными функциями программа позволяет:

•копировать и восстанавливать выборочные файлы и группы файлов;
•инициировать диски до начала операций;
•управлять работой лент (перемотка, проверка меток и т. д.);
•выдавать вспомогательную информацию и выполнять другие сервисные функции.

Командная строка программы BRU отличается от командных строк других утилит и может иметь длину до 256 символов, включая строки продолжения. Основной формат командной строки:

BRU>/ключи устр. ввода [: ,... [спец.ф, ...] устр. выв1: ,...

Если командная строка имеет вид:

BRU>/ключи <CR>

то на терминале появляются следующие подсказки:

```
FROM:
TO:
INITIALYZE [Y/N]
```

FROM:—в ответ на эту подсказку вводятся имена устройств, на которых размещены вводные наборы данных в соответствующем формате;

TO: — требует ввода имен выводных устройств в соответствующем формате;

INITIALYZE [Y/N] — задается вопрос, инициализировать ли выводной том.

Если в качестве устройства ввода (вывода) используются магнитные ленты, то может быть задано более одного устройства (MT0:, MT1:). В командной строке может быть указано до 16 спецификаций файлов. Можно копировать или восстанавливать файлы по именам, каталогам и т.д. (см. разд. 6.2.1). Для продолжения командной строки более чем на одну строку в качестве символа продолжения используется дефис (-).

О с н о в н ы е к л ю ч и :

/APP[END] — заставляет добавить набор копируемых файлов вводного тома к последнему из файлов на выводном томе.

/BAC[KUP_SET] — указывает имя копируемого на ленту набора, по умолчанию берется имя дискового тома.

/COM [PARE] — заставляет сравнивать данные на вводном и выводном устройстве и сообщать о всех различиях.

/CRE [ATED]: [BEF[ORE], AFT [ER]]: (дата: время) — копирует файлы, созданные до (BEF) или после (AFT) указанно даты. Дата и время (в круглых скобках) вводятся, как в команде TIME MCR (см. разд. 4.2).

/DEN[SITY]: значение — задает плотность записи на магнитной ленте: Допустимые значения:

для MT: — 800. = 32 бит/мм (по умолчанию);

для MM: — 800. = 32 бит/мм (по умолчанию) или 1600.= 64 бит/мм (по выбору).

/DIR[ECTORY] — распечатывает имена скопированных наборов или файлов на магнитной ленте.

/DIS[PLAY] — распечатывает на терминал имена файлов и их коды UFD.

/ERR[ORS]: число — прекращает операцию восстановления после возникновения указанного числа ошибок при чтении с ленты.

/EXC[LUDE] — исключает из операции копирования или восстановления файлы, заданные в командной строке.

/EXT[END] — задает число блоков, на которое файл может быть расширен при превышении выделенного ему пространства.

/HEA[DERS]: число — задает количество заголовков, под которые реализуется пространство при создании индексного файла.

/INI[TIALYZE] — служит для инициации выводного дискового тома, при этом на томе размещается файловая структура ОСРВМ.

/INV[OLUME]: имя — задает метку тома вводного носителя, имя не может содержать более 12 символов.

/MOU[NTED] — позволяет копировать и восстанавливать файлы на томах, которые монтированы как тома с файловой структурой ОСРВМ.

/NEW[VERSION] — разрешает конфликты в спецификациях файлов, которые возникают во время операции восстановления, созданием новых версий файлов.

/NOI[NITIALYZE] — указывает, что не нужно инициализировать диск, так как он уже имеет файловую структуру ОСРВМ.

/NOS[UPERSEDE] — указывает, что при восстановлении данных на монтированный диск, при идентичности спецификаций файлов, информация с вводного диска не передается.

/OUT[VOLUME]: имя — задает метку тома выводного диска.

/POS[ITION]: [BEGINNING, MIDDLE, END, BLOCK: N] — определяет положение индексного файла на томе при инициализации диска. Параметры BEGINNING, MIDDLE, END задают соответственно начало, середину и конец тома, а параметр BLOCK — номер блока, с которого начинается индексный файл.

/PRO[TECTION]: код защиты — задает умолчание кодов защиты для файлов, создаваемых на томе

после завершения программы BRU (см. ключ /PR в разд. 6.2.1).

/REV[ISED]: [BEFORE, AFTER]: (дата время) — обеспечивает копирование или восстановление файлов, измененных до или после указанной даты и времени. Аргументы используются такие же, как и в ключе /CRE. Если определяется только дата или время, круглые скобки не нужны.

/REW[IND] — перематывает вначале первую магнитную ленту из последовательности перед выполнением копирования восстановления.

/SUP[ERSEDE] — позволяет в случае совпадения спецификаций файлов по этому ключу файл на выводном томе заменить его файлом с вводного тома.

/TAPE[LABLE]: метка — задает 6-символьный индентификатор в метке тома на магнитной ленте.

/UFD — создает каталог UFD на монтированном выводном томе.

/VER[IFY] — после копирования данных сравнивает их и сообщает различия.

/WIN[DOWS]: число — при инициализировании выводного диска задает умолчание для числа указателей извлечений для файла.

7.4.2. ПРОГРАММА СОХРАНЕНИЯ И УПЛОТНЕНИЯ ТОМА DSC

Программа сохранения и уплотнения тома копирует файлы, содержащиеся на диске с файловой структурой ОСРВМ, на ленту или на диск или с ленты формата DSC на диск. Файлы и их расширения размещаются в смежных блоках, что сокращает требуемое число указателей извлечений и заголовков файлов на создаваемом диске.

Файлы, ранее расположенные произвольно, переписываются на новый накопитель плотно, без промежуточного пространства, а имеющееся свободное пространство объединяется в одну последовательную область.

Программа DSC, работающая под управлением системы, запускается двумя способами:

<DSC командная строка

или

<DSC <CR>

Системно-независимая версия программы DSC запускается по команде оператора BOO. Ф о р м а т :

выв. устр..: метка файла/ключ = вв.устр.: метка файла/ключ

где выв. устр..: — список физических устройств, на которые передаются данные;
вв. устр.: — список физических устройств, с которых копируются данные;
метка файла — идентифицирует метку тома диска или метку файла на ленте;
/ключ — один или более ключей DSC.

К л ю ч и :

AP — служит для записи файлов на ленточный том, который содержит ранее созданный DSC файл (файлы). Этот ключ используется с выводной спецификацией файла.

BAD — применяется с выводным диском для введения информации о дефектных блоках. Этот ключ позволяет дополнять или игнорировать файл дефектных блоков, а также использовать только вручную введенную информацию о дефектных блоках. Ф о р м а т :

$$\text{BAD=} \begin{cases} \text{MAN} \\ \text{NOAUTO} \\ \text{OVR} \\ \text{MAN:NOAUTO} \\ \text{MAN:OVR} \end{cases}$$

где MAN — позволяет вручную вводить дополнительные номера дефектных блоков в файл BADBLK.SYS;
NOAUTO — игнорирует описатель дефектных блоков на диске;
OVR — игнорирует описатель дефектных блоков на диске с последней дорожкой и использует последний годный к применению блок на диске, исключая последнюю дорожку;
MAN : NOAUTO — вводит в файл BADBLK.SYS только вручную введенные номера дефектных блоков;
MAN : OVR — позволяет вручную дополнять файл BADBLK.SYS, для которого употребляется описатель из последнего годного к использованию блока диска, исключая последнюю дорожку.

Если заданы значения MAN, MAN : NOAUTO, MAN : OVR, программа DSC отвечает подсказкой:

DS>BN(S) =

В ответ на эту подсказку требуется ввести числа в формате N : M, где N — логический номер

начального блока группы дефектных блоков, а M — число последовательных блоков в этой группе.

Если задается несколько последовательностей дефектных блоков в данной строке, то они разделяются запятой, пробелом или знаком табуляции.

/BL = N или /BL: N — позволяет установить число блоков, передаваемых за одну операцию ввода-вывода (по умолчанию — 4). Максимальное значение N для системы с ДП — 36 (10), а для систем без ДП — 10 (10).

/CMP — указывает на необходимость сравнения двух дисков, диска и набора лент или набора лент и диска. /CMP задается как часть выводной спецификации и используется для сравнения содержимого томов без обращения к операции копирования.

/DENS — по своему действию аналогичен ключу /DEN программы BRU.

/RW — служит для перемотки каждой ленты из набора до выполнения любой операции DSC. Если /RW используется во вводной спецификации, то перематывается первая лента из набора до начала операции копирования, другие ленты перематываются по мере необходимости. Если ключ /RW определен вместе с меткой файла, то после перемотки ленты программа DSC ищет указанный файл. Если ключ /RW употребляется в выводной спецификации, то перемотка ленты происходит до операции копирования или сравнения.

/VE — служит для проверки содержимого томов после копирования. Этот ключ используется с дисками в выводной части командной строки.

Для системно-независимой версии программы DSCSYS используются дополнительные ключи. Употребление ключей /VEC и /CSR аналогично их применению системно-независимой версией программы BAD.

/UNIT = N — изменяет номер устройства на номер, допустимый в программе DSCSYS. Системно-независимой версии программы доступны устройства с логическими номерами 0, 1. Ключом /UNIT указывается фактический номер устройства N, с которым будет работать программа DSCSYS.

П р и м е р :

```
DS>DPl : /UNIT = 4
DS>DP1 : =DP0:
```

Данные будут копироваться с устройства DP0: на DP1:, которому назначено устройство с номером 4.

7.5. ОТЛАДОЧНЫЕ ПРОГРАММЫ

В состав отладочных программ, рассматриваемых в данном подразделе, входят программа-отладчик (ODT), программа изменения объектного модуля (PAT) и программа изменения файлов образа задачи (ZAP).

7.5.1. ПРОГРАММА-ОТЛАДЧИК ODT

Программа ODT позволяет отлаживать оттранслированные программы. Работая в режиме диалога с отладочной программой, пользователь может:

•напечатать содержимое любой ячейки в объектной программе для просмотра или изменения;

•выполнить всю или некоторую часть объектной программы, используя точки останова;

•найти слова в объектной программе по указанным двоичным образам;

•найти в объектной программе инструкции, которые ссылаются на указанный адрес;

•вычислить смещения адресов относительно счетчика инструкций и смещения переходов внутри объектной программы;

•заполнить указанный пользователем блок слов или байтов нужными значениями;

•распечатать указанный пользователем блок слов или байтов для исследования. Во время построения задачи программа-отладчик компонуется совместно с программой пользователя (ключ /DA описан в разд. 5). Программа ODT является монитором задачи пользователя и выполняется с ее привилегиями.

Для упрощения дальнейшего изложения введем некоторые понятия и определения.

Точки останова — указанные пользователем адреса, при достижении которых выполнение программы должно быть приостановлено для взаимодействия программиста с задачей, т. е. для просмотра и исправления ячеек памяти, поиска переменных и т. д. Задача после запуска выполняется до тех пор, пока не встретится точка останова, в которой управление передается отладчику; он выдает подсказку и ждет ввода команды пользователя. Программа ODT допускает одновременно до 8 точек

останова.

Регистры смещения — ячейки отладчика, содержащие абсолютные базовые адреса объектных модулей после их размещения в памяти. Транслятор, создавая объектный модуль, считает равным нулю базовый адрес каждой программной секции. После компоновки модулей большинство значений внутри них изменяется на постоянную величину, которая является базовым адресом или относительным смещением объектного модуля. В процессе отладки эти смещения нужно постоянно вычитать из абсолютных адресов, чтобы установить соответствие между адресами модуля и листинга. Программа ODT автоматически вычисляет смещение каждого объектного модуля благодаря наличию 8 регистров смещения ($R0— $R7), в которые записываются относительные смещения модулей. Смещение каждого модуля берется из карты памяти MAP и заносится в соответствующий регистр смещения.

Открытая ячейка — слово или байт, содержимое которых напечатано для просмотра и может быть изменено. Для открытия ячейки нужно указать ее адрес и одну из команд открытия.

Для определения типов выражения адреса введем некоторые обозначения:

a — аргумент, используемый для указания адреса ячейки задачи;

n — восьмеричная цифра от 0 до 7;

k — восьмеричное число из 6 цифр (*k* не превышает 177777) или выражение, которое приводится к этому числу.

Адрес открытой ячейки может определяться следующим образом:

k — значением адреса *a* является значение *k*; *n*, *k* — значением *a* является значение *k* плюс содержимое регистра смещения с номером *n* (*n* принимает значение от 0 до 7). В выражении адреса тоже может использоваться предварительно заполненный регистр константы С.

Ниже представлены форматы знаков и символов, являющихся командами и операторами программы-отладчика.

Формат	Значение
+или пробел	Арифметический оператор. Сумма предшествующего аргумента и следующего за ним образует текущий аргумент
–	Арифметический оператор. Вычитание следующего аргумента из предыдущего образует текущий аргумент
–	Оператор регистра смещения. Использует предыдущую восьмеричную цифру для ссылки к одному из восьми регистров смещения отладчика и берет содержимое этого регистра и значение аргумента, следующего за запятой, чтобы образовать текущий аргумент
*	Эта команда используется в образовании аргументов RADIX-50.
(точка)	Оператор текущей ячейки. Вызывает адрес последней явно открытой ячейки, который должен быть использован как текущий адрес для операций отладчика — адрес, предполагаемый командой ^ для возврата к предыдущей последовательности открытых ячеек. Этот адрес подразумевается также в использовании команд: /' (апостроф), « (кавычки), %, <LF>
; (точка с запятой)	Разделитель аргументов. Разделяет аргументы, используется при формировании выражения адреса или значения регистра программы-отладчика
<CR> или K<CR>	Закрывает текущую открытую ячейку. В формате K<CR>, K<LF>, K:, K←, K@, K>, K< значение K заменяет содержимое текущей открытой ячейки, прежде чем она будет закрыта
<LF> или K<LF>	Закрывает открытую в настоящее время ячейку, открывает следующую за ней ячейку, т. е. печатает ее содержимое
: или K	Закрывает открытую ячейку и печатает содержимое предшествующей ячейки (на клавиатурах некоторых устройств вместо знака **:** используется знак **^)**
← или K←	Берет содержимое открытой ячейки как смещение счетчика адреса и вычисляет адрес следующей ячейки, которая должна быть открыта; закрывает открытую текущую ячейку и печатает содержимое новой ячейки по вычисленному адресу (на клавиатуре некоторых устройств вместо знака ← используется знак **–)**
@ или K@	Берет содержимое открытой ячейки как абсолютный адрес, закрывает текущую ячейку и печатает содержимое ячейки по этому абсолютному адресу
> или K>	Берет младший байт открытой ячейки как смещение относительного перехода к адресу следующей ячейки, которая должна быть открыта. Вычисляет этот адрес, закрывает открытую ячейку и печатает содержимое ячейки по вычисленному адресу. Вычисление адреса происходит следующим образом: берется младший байт открытой ячейки, умножается на 2, к результату прибавляется 2 и полученная сумма прибавляется к адресу текущей ячейки
< или K<	Закрывает ячейку и заново открывает последнюю явно открытую ячейку, предшествующую последней из команд ←,@, <
$*n*	Адрес одного из восьми универсальных регистров программы, где *n* — номер регистра от 0 до 7
$X или $*n*X	Адрес одного из 16 типов специальных внутренних регистров отладчика. В формате $*n*X

восьмеричная цифра *n* определяет номер регистра внутри группы данного типа; X является одним из следующих алфавитных указателей регистров:

S — регистр состояния процессора (обозначается PS), который охраняется отладчиком, когда появляется точка останова или ошибка программы пользователя;
W — регистр слова состояния директивы $DSW задачи;
A — регистр аргумента поиска;
M — регистр маски предела памяти;
L — регистр нижнего предела памяти;
H — регистр верхнего предела памяти;
C — регистр константы;
Q — регистр величины;
F — регистр формата;
X — векторный регистр повторной входимости;
*n*B — регистры адреса точки останова;
*n*G — регистры счетчика продолжения с точек останова;
*n*L — регистры инструкций точек останова.

*n*R	Регистры смещения
*n*V	Векторные регистры синхронного системного прерывания
*n*E	Регистры содержимого стека синхронного системного прерывания
*n*D	Регистры логического номера устройства
C	Оператор регистра константы. Представляет содержимое специального регистра $C
Q	Оператор регистра величины. Представляет содержимое специального регистра $Q
" или *a"*	Оператор вывода слов в символьном коде. Интерпретирует в символьном коде и печатает содержимое текущей открытой ячейки как два символа в символьном коде и запоминает это слово в регистре величины $ Q. В формате *a"* значение а берется как адрес ячейки, содержимое которой должно быть интерпретировано и распечатано
' или *a'*	Оператор вывода байтов в символьном коде. Интерпретирует и печатает содержимое текущей или последней открытой ячейки как один символ в символьном коде и запоминает это слово в регистре величины $Q. В формате *a'* значение *a* берется как адрес ячейки, которая должна быть интерпретирована и распечатана
% или *a*%	Оператор вывода слов в коде RADIX-50. Интерпретирует и печатает содержимое открытой или последней» открытой ячейки как три символа в коде RADIX-50 и запоминает это слово в регистре величины $Q. В формате *a*% значение *a* берется как адрес ячейки, которая должна быть интерпретирована и распечатана
/ или *a*/	Оператор вывода восьмеричных слов. Распечатывает содержимое последнего открытого слова и запоминает это содержимое в регистре величины $Q. В формате *a/* значение о берется как адрес слова, которое должно быть открыто
\ или *a*\	Оператор вывода восьмеричных байтов. Распечатывает содержимое последнего открытого байта и запоминает это содержимое в регистре величины $Q. В формате *a*\ значение *a* берется как адрес байта, который должен быть открыт
K=	Представляет выражение *K* в виде 6-значного восьмеричного числа, печатает его и запоминает в регистре величины
8 или 9	Отменяет текущую команду. Десятичные цифры 8 или 9 являются недопустимыми знаками. Появление 8 или 9 заставляет отладчик игнорировать текущую команду, поэтому для отмены текущей команды достаточно напечатать 8 или 9
B	Удаляет все точки останова из задачи пользователя
*n*B	Удаляет *n*-ю точку останова из задачи пользователя
a; B	Устанавливает следующую последовательную точку останова в задаче пользователя по адресу *a*
a; *n*B	Устанавливает *n*-ю точку останова по адресу *a* в задаче пользователя
K	Используя этот регистр смещения, содержимое которого меньше или равно адресу текущей открытой ячейки, вычисляет разность (смещение в байтах) между адресом текущей открытой ячейки и величиной, содержащейся в выборочном регистре смещения, печатает это смещение и запоминает его в регистре величины
nK	Вычисляет смещение (в байтах) между адресом текущей открытой и последней открытой ячейки и величиной, содержащейся в регистре смещения *n* печатает это смещение и запоминает его в регистре величины
a, *nK*	Вычисляет смещение (в байтах) между адресом *a* и величиной, содержащейся в регистре смещения ***n,*** печатает это смещение и запоминает его в регистре величины
F или KF	Заполняет область памяти, заданной регистрами пределов памяти ($L или $H), значением, указанным в регистре $/A, или значением *K*
G или *a*G	Устанавливает инструкции BPT или восстанавливает инструкции BPT по всем ячейкам точек останова в образе задачи, восстанавливает слово состояния процессора и программные регистры пользователя, затем начинает выполнение с адреса, указанного в счетчике инструкций программы пользователя ($7). В формате *a*G значение *a* заменяет текущее содержимое седьмого регистра, прежде чем начнется выполнение задачи
*a*O или *a*; KO	Вычисляет и печатает смещение от адреса текущей открытой ячейки 2 до адреса *a* и 8-

	разрядное смещение перехода. Команда *a*; КО вычисляет и печатает смещение относительно PC и смещение относительно перехода от адреса *a* до адреса *K*
KP или *P*	Продолжает выполнение программы пользователя от текущей точки останова и останавливает выполнение, когда встретится следующая точка останова или когда задача завершится. В формате *KP* продолжается выполнение программы от текущей точки останова и останавливает выполнение программы в точке останова при прохождении через нее в *K-й* раз
R	Устанавливает все регистры смещения в —1, т. е. максимальное значение адреса равно 177777 (8)
*n*R	Устанавливает в *n*-й регистр смещения —1, т. е. максимальное значение адреса равно 177777 (8)
a;R	Устанавливает следующий последовательный регистр смещения на значение адреса *a*
a; *n*R	Устанавливает регистр смещения *n* на значение адреса *a*
S или *n*S	Выполняет одну или *n* инструкций и печатает адрес следующей инструкции, которая должна быть выполнена
W или KW	В границах памяти, указанных в регистре нижнего предела памяти $L или в регистре верхнего предела памяти $H, отыскивает слова по указанному аргументу поиска, содержащемуся в регистре аргумента поиска $A. Каждое слово памяти и аргумент поиска сравнивается по маске, указанной в регистре маски поиска **$**M
m; W или *m*; *k*W	В случае совпадения печатается адрес соответствующей ячейки и ее содержимое. В формате *k*W значение *k* заменяет текущее содержимое $A перед началом поиска. В формате *m*; W значение *m* заменяет текущее содержимое $M. В формате *m*; *k*W значение *k* и *m* заменяет текущее содержимое соответственно @A и $M
E или *k*E или *m*; E или *m*; *k*E	В границах памяти, указанных в регистре нижнего предела памяти $L и в регистре верхнего предела памяти $H, осуществляет поиск слов, которые ссылаются на исполнительный адрес, указанный в регистре аргумента поиска $A, который маскируется значением, указанным в регистре маски $M. Эти слова могут быть исполнительным или относительным адресом либо смещением относительного перехода к исполнительному адресу, указанному в $A. В формате *k*E значение *k* заменяет текущее содержимое $A перед началом поиска. В формате *m*; E значение *m* заменяет текущее содержимое $M перед началом поиска. В формате *m*; *k*E значения *m* и *k* заменяют текущее содержимое регистров соответственно $A и $M
L или KL или *a*; L или *a*; *k*L или *n*; *a*; *k*L	Распечатывает все ячейки в границах памяти, указанных в регистре нижнего предела памяти $L или в регистре верхнего предела памяти $H, используя устройство печати, указанное в регистре логического номера устройства $ID. В формате *k*L значение *k* заменяет текущее содержимое $H перед началом операции. В формате *a*; L значение a заменяет текущее содержимое $L перед началом операции. В формате *a*; *k*L значения a и *k* заменяют текущее содержимое соответственно $L и $H. В формате *n*; *a*; *k*L значение *n* указывает регистр $ND, содержащий логический номер устройства, на которое должна быть введена распечатка
F или *k*F	Заполняет содержимым регистра аргумента поиска | A ячейки памяти внутри пределов памяти, указанных в регистре нижнего предела памяти $L и в регистре верхнего предела памяти $H. В формате *k*F значение *k* заменяет текущее содержимое регистра $A, перед тем как начнется операция заполнения
V	Разрешает обращение из программы ODT ко всем векторам SST и внесение адресов входных точек ловушек ODT в таблицу, используемую директивой управляющей программы SVDB $
X	Означает выход из отладчика и возврат к управляющей программе ОСРВМ

П р и м е ч а н и е . При поиске под маской производится сравнение слов памяти и аргумента только в тех разрядах, которые установлены в 1 в маске; все другие разряды игнорируются. В границах памяти, указанных в регистре нижнего предела памяти $L и в регистре верхнего предела памяти $H, осуществляется поиск слов, которые не соответствуют аргументу поиска, указанному в регистре аргумента поиска $A.

7.5.2. ПРОГРАММА ИЗМЕНЕНИЯ ОБЪЕКТНОГО МОДУЛЯ PAT

Программа PAT позволяет изменять или корректировать коды в перемещаемом двоичном модуле. Программа PAT считывает файл, содержащий корректирующую информацию, и обрабатывает на основе этой информации объектный модуль. Данные корректировки готовятся в исходном виде, а затем транслируются макроассемблером.

Программа PAT вызывается одним из методов, описанных в разд. 7.1.

Ф о р м а т командной строки PAT:

[выв. ф] = вв. ф [/CSE [:число]], файл корректировки [/CS [:число]]

где выв. ф — спецификация выводного файла, если она не указана, то выводной файл не создается;

вв. ф — спецификация вводного файла, который может содержать один или более объектных модулей;

файл корректировки — спецификация файла корректировки, который содержит корректировку одного модуля из вводного файла;

/CS — ключ контрольной суммы, по которому создается восьмеричное значение суммы всех двоичных данных файла;

число — восьмеричное значение, в котором программа PAT сравнивает вычисленное значение контрольной суммы.

PAT выполняет корректировку относительно базы вводного модуля, используя дополнения и исправления из файла корректировки. Во время работы программы PAT обнаруженные в файле корректировки новые глобальные символы добавляются в таблицу символов модуля. Глобальные символы в файле корректировки, совпадающие с глобальными символами вводного файла, заменяют их, определяя эти символы как относительные либо как абсолютные.

7.5.3. ПРОГРАММА ИЗМЕНЕНИЯ ФАЙЛОВ ОБРАЗА ZAP

Обслуживающая программа ZAP позволяет проверять и изменять файлы образов задач и данных на томе с файловой структурой ОСРВМ. Работа программы ZAP во многом похожа на работу программы-отладчика, поэтому предварительно рекомендуем ознакомиться с разд. 7.5.1. Программа ZAP позволяет:

•распечатывать и изменять слова и байты в образах задач на диске;

•распечатывать дисковые блоки и адреса сегментов задач. При этом программа ZAP использует ключи, односимвольные команды и специальные внутренние регистры (аналогично отладчику ODT). Программа ZAP может быть вызвана любым способом из перечисленных в разд. 7.1. Однако наиболее употребимый следующий:

```
>ZAP
```

или

```
2AP>спецификация файла [/ключ]
```

К л ю ч и :

/AB — обрабатывает адреса внутри файла. Этот ключ указывается обязательно, если файл не является образом задачи.

/RO — обеспечивает выполнение функций ZAP без корректировки информации на диске.

/LI — выдает начальный блок файла и граничные адреса для каждого сегмента перекрытия в следующем виде:

```
ssssss: aaaaaa-bbbbbb
```

где ssssss — указывает начальный блок в восьмеричном виде;
aaaaaa — указывает нижнюю границу адреса в восьмеричном виде;
bbbbbb — указывает верхнюю границу адреса в восьмеричном виде.

Режимы адресации в образе задачи. Как было показано в разд. 7.5.1, адреса листинга ассемблера не соответствуют фактическим адресам в образе задачи, в котором большинство значений смещено на константу, являющуюся абсолютным базовым адресом программной секции после его размещения. В противном случае эта константа называется *относительным смещением программной секции.* Для приведения в соответствие адресов листинга с адресами в образе задачи используются регистры смещения 0R — 7R.

Программа ZAP имеет два режима адресации к образу задачи: абсолютную и адресацию относительного образа задачи.

При абсолютной адресации (с ключом /AB) программа ZAP интерпретирует первый адрес в файле как сегмент 1 с начальным адресом 000000, который используется как базовый при интерпретации других вводимых адресов. При этом режиме можно обращаться ко всем байтам файла, в частности к блокам заголовка образа задачи. Этот способ обычно применяется для односегментных задач.

В способе адресации относительно образа задачи программа ZAP позволяет адресоваться к ячейкам, используя номер блока и относительные смещения, которые берутся из карты распределения памяти построителя задач.

Команды программы ZAP. Существуют три группы команд ZAP: команды открытия и закрытия ячеек, основные команды и возврат каретки.

К о м а н д о й о т к р ы т и я - з а к р ы т и я я ч е е к служит введение одного из следующих символов: /, », %, \, ', ! или ^ , ← или —, @, >, < . Эти команды полностью соответствуют одноименным командам ODT (см. разд. 7.5.1).

К о с н о в н ы м относятся команды = и R, которые работают так же, как в программе ODT, а также команда V, которая проверяет содержимое текущей ячейки.

В о з в р а т к а р е т к и <CR> заставляет закрыть текущую измененную ячейку. Два последовательно введенных символа <CR> открывают следующую ячейку в файле. В отличие от ODT

команды ZAP действуют только после символа <CR>.

Программа ZAP использует специальным образом несколько фиксированных ячеек памяти, называемых *внутренними регистрами.* Имеется 8 регистров смещения (0R — 7R), постоянный регистр, или регистр константы C, регистр формата F, регистр величины Q. Использование этих регистров, арифметических операторов (+, —, ∗) и символов-разделителей командной строки (; ,) полностью совпадает с их применением программой ODT.

7.6. ПРОГРАММА-БИБЛИОТЕКАРЬ LBR

Обслуживающая программа-библиотекарь позволяет создавать, корректировать, модифицировать и распечатывать библиотечные файлы — наборы данных прямого доступа; содержащие файлы одного типа.

Библиотечные файлы содержат две справочные таблицы: таблицу входных точек (EPT) и таблицу имен модулей (MNT). Таблица EPT включает имена точек входа, которые в исходных программах определены как глобальные символы, а таблица MNT — имена модулей в библиотеке.

Существуют три типа библиотек: объектная (.OLB), макро (.MLB) и универсальная (.ULB).

О б ъ е к т н ы е библиотеки состоят из объектных модулей. Программа-библиотекарь обращается к модулям по имени, а ссылки к ним из исходных программ осуществляются по именам точек входа. Объектные библиотеки используются построителем задач как входные данные.

М а к р о б и б л и о т е к и включают наборы макрокоманд в исходном виде. Имена модулей берутся из директив MACRO. Макробиблиотеки используются макроассемблером в качестве входных данных.

У н и в е р с а л ь н ы е библиотеки содержат модули, состоящие из любых файлов одинакового типа программ или текстов. Из программы к таким модулям можно ссылаться, используя подпрограмму $ULA, находящуюся в системной библиотеке (SYSLIB.OLB).

Программа LBR может вызываться любым способом, описанным ранее (см. разд. 7.1).

Ф о р м а т командной строки программы-библиотекаря:

выв. ф [, файл листинга] — вв. ф1 [, вв. ф2, ..., вв. ф N]

К л ю ч и программы LBR:

/CO — создает новую версию библиотечного файла, физически удаляя все логически вычеркнутые записи, сдвигая свободное пространство в конец файла и делая его пригодным для включения новых модулей. С помощью десятичных числовых ключей можно изменить Характеристики файла. Ф о р м а т :

выв.ф/CO [:РБФ] [:РТВХ] [: РТИМ] = вв.ф

где РБФ — размер нового библиотечного файла в 256-словных блоках. По умолчанию размер старого файла;

РТВХ — число записей, которые можно разместить в таблице входных точек (не более 4096). По умолчанию берется значение из старой библиотеки;

РТИМ — число записей в таблице имен модулей (не более 4096). По умолчанию берется значение из старой библиотеки. Числа РТВХ и РТИМ берутся кратными 64.

/CR — создает непрерывный библиотечный файл на диске, в том числе заголовок файла и соответствующие таблицы. Ф о р м а т к л ю ч а :

выв.ф/CR:РБФ:РТВХ:РТИМ:ТИПБ:ТИПВФ

где параметры РБФ, РТВХ, РТИМ имеют такое же значение, что и у ключа /CO. По умолчанию параметр РБФ равен 100, РТВХ— 152, РТИМ — 256 (OBJ — для объектных библиотек, MAC — для макробиблиотек и UNI — для универсальных);

ТИПБ — тип создаваемой библиотеки;

ТИПВФ — умолчание типа файла для создаваемой универсальной библиотеки, если параметр не задан, то по умолчанию UNI.

/DE — производит логическое удаление библиотечных модулей и их входных точек. При логическом удалении модуль становится недоступен, однако занимает место в библиотеке. Ф о р м а т к л ю ч а :

выв.ф/DE: МОД1, МОД2..... МОД15

где МОД — имя удаляемого модуля.

/DF — определяет умолчание типа библиотечного файла. Ф о р м а т к л ю ч а :

выв. ф/DF :ТИП или /DF :ТИП 288

Допустимые значения параметров такие же, как в параметре ТИПБ ключа /CR.

/DG — служит для удаления входных точек из таблицы EPT. Причем этот ключ не удаляет глобальный символ из модуля, который содержит его фактическое определение. Ф о р м а т к л ю ч а :

выв.ф/DG :ГЛОБЛ1:ГЛОБЛ2: ... :ГЛОБЛ15

/EP — предоставляет возможность включить или исключить запись глобальных символов в таблицу входных точек библиотеки. Если указывается положительное значение ключа /EP, то входные точки включаются в таблицу EPT, если отрицательное /— EP или /NOEP, то не включается. Ф о р м а т :

выв.ф [/EP] = вв.ф[/EP], ..., вв.ф[/EP]

Если /EP стоит в спецификации выводного файла, то программа-библиотекарь предполагает, что каждый вводной файл содержит модули с включаемыми или исключаемыми входными точками. Если /EP указан в спецификации вводного файла, то для модулей только этого файла определяются, включаются или исключаются их входные точки. Ключ /EP служит умолчанием для программы-библиотекаря.

/EX — позволяет извлекать из библиотечного файла один или несколько модулей и писать их в указанный выводной файл. Библиотечный файл остается без изменений. Для объектной и макробиблиотеки может быть указано до 8 модулей, для универсальной — только один. Ф о р м а т :

выв.ф = вв. ф/EX [:МОД1:...:МОДN]

/IN — предоставляет возможность вставлять модули в библиотечный файл. Для макробиблиотек из вводных файлов извлекаются только макроопределения верхнего уровня. Ф о р м а т к л ю ч а :

выв.ф [/IN]=вв.ф1, [вв.ф2,..., вв.фN]

Для универсальных библиотек ключ /IN указывается в спецификации вводного файла. После ключа можно указать имя модуля и описательную информацию, которая сохраняется в заголовке модуля. Ф о р м а т :

выв. ф = вв. ф/IN [:МОД:ОП:ОП:ОП:ОП]

/LI, /LE, /FU — позволяет распечатывать оглавление библиотечного файла;

/LI — распечатывает имена всех модулей библиотеки;

/LE — распечатывает имена всех модулей библиотеки и их входных точек;

/FU — распечатывает имена всех модулей библиотеки и дает полное описание каждого модуля (размер, дата включения и т. д.). Ф о р м а т :

выв. ф [, файл листинга] /ключи

/MH — относится только к универсальным библиотекам и позволяет изменять задаваемые пользователем заголовки в описателе модуля. Ф о р м а т :

выв. ф/MH:МОД:ОП:ОП:ОП:ОП

/RP — заменяет модули в библиотечном файле на вводные модули с теми же именами. Для макробиблиотек извлекаются макроопределения только первого уровня. Ранее существующий модуль удаляется логически, а из таблицы EPT удаляются входные точки. Ключ замены может быть указан в локальном или глобальном формате. В глобальном формате ключ /RP присоединяется к спецификациям выводного файла и предполагается, что все вводные файлы содержат модули для замены. В локальном формате ключ /RP присоединяется к спецификации вводного файла и считается, что только файл с ключом /RP содержит модули для замены. Ф о р м а т :

выв.ф [/RP] = вв.ф1 [/RP] [,вв.ф2 [/RP], ..., вв.фN [/RP]

Используя ключ /RP для универсальных библиотек, можно, как и для ключа /IN, в командной строке указать имя модуля и описательную информацию.

/SP — функционирует так же, как и в программе PIP (см. разд. 7.2.1).

/SS — используется при операциях вставки для установки бита выборочного поиска в байте атрибутов заголовков объектных модулей при включении их в объектную библиотеку. Объектные модули с атрибутом выборочного поиска обрабатываются построителем задач специальным образом. Глобальные символы, определяемые в таких модулях, включаются построителем задач в таблицу символов (STB) в том случае, если на них есть ссылки из других задач. Ф о р м а т :

выв.ф = вв.ф1 /SS [,вв.ф2 [/SS], ..., вв.фN [/SS]

/SZ — уменьшает размеры макрораспределений, удаляя из них пробелы, знаки табуляции, пустые строки и комментарии. Ключ сжатия употребляется в локальном и глобальном формате (отличия форматов такие же, как и в ключе /RP). Ф о р м а т :

выв. в [/SZ]=вв.в1 [/SZ] [,вв.в2 [/SZ], ..., вв.вN [/SZ]]

1. Коды завершения

В данном приложении приводятся числовые значения и символические имена кодов завершения директив управляющей программы, возвращаемые задаче в слове DSW.

Код завершения директивы может быть проверен задачей как по символическому имени, так и по числовому значению. Значения символических имен кодов завершения по умолчанию определяются на этапе построения задачи. Символические имена кодов завершения на этапе трансляции программы, написанной на языке программирования макроассемблере определяются макрокомандой DRERR$. В программу должны быть включены следующие строки:

```
.MCALL DRERR$
DRERR$
```

Список стандартных кодов завершения директив приведен в табл. П.1.

Таблица П.1

Символическое имя	Числовое значение	Значение
IS.CLR	+00	Флаг события «обнулен»
IS.SUC	+01	Успешное завершение директивы
IS.SET	+02	Флаг события установлен
IE.UPN	—01	Недостаточно системной динамической памяти
IE.INS	—02	Указанная задача не установлена
IE.PTS	—03	Район мал для задачи
IE.UNS	—04	Недостаточно системной динамической памяти для посылки данных
IE.ULN	—05	Неназначенный LUN
IE.HWR	—06	Драйвер устройства нерезидентен
IE.ACT	—07	Задача не активна
IE.ITS	—08	Директива не соответствует состоянию задачи
IE.FIX	—09	Задача фиксирована (не фиксирована)
IE.CKP	—10	Задача, издавшая директиву, невыгружаемая
IE.TCH	—11	Задача выгружаемая
IE.RBS	—15	Буфер для получения данных мал
IE.PRI	—16	Нарушение привилегий
IE.RSU	—17	Ресурсы заняты
IE.NSW	—18	Нет пространства для выгрузки
IE.ILV	—19	Указанный вектор недопустим
IE.ITN	—20	Неверный номер таблицы
IE.LNF	—21	Логическое имя не найдено
IE.AST	—80	Директива используется (не используется) из AST-программы
IE.MAP	—81	Программа обработки прерываний или программа разрешения (запрещения) прерываний не в границах 4 Кслов, считая от значения < базовый адрес & 177700>
IE.IOP	—83	В виртуальном окне выполняется требование ввода-вывода
IE.ALG	—84	Ошибка выравнивания границы адресов
IE.WOV	—85	Ошибка наложения адресных окон при создании адресного окна
IE.NVR	—86	Недопустимый ID района
IE.NVW	—87	Недопустимый ID адресного окна
IE.ITP	—88	Недопустимый параметр для TI:
IE.IBS	—89	Недопустимый размер буфера при посылке данных (>255.)
IE.LNL	—90	LUN блокирован от использования
IE.IUI	—91	Недопустимый UIC
IE.IDU	—92	Недопустимое имя устройства или недопустимый номер устройства
IE.ITI	—93	Недопустимые параметры времени
IE.PNS	—94	Раздел (район) не существует в системе
IE.IPR	—95	Недопустимый приоритет (>250)
IE.ILU	—96	Недопустимый LUN
IE.IEF	—97	Недопустимый номер флага (EFN>64)
IE.ADP	—98	Часть блока DPB расположена вне памяти задачи, издавшей директиву
IE.SDP	—99	Недопустимый код директивы или размер блока DPB

2. Коды идентификации директив

Коды идентификации директив (DIC) используются для идентификации директив управляющей программы. Код размещается в младшем байте первого (возможно, единственного) слова блока DPB. Длина блока DPB в словах размещается в старшем байте первого слова блока DPB. Первое слово блока DPB имеет следующий формат:

старший байт: длина блока DPB младший байт: DIC

В табл. П.2 приводятся значения первых слов блока DPB системных директив, упорядоченных по числовому значению. Эти значения полезны при отладке программ, позволяя по восьмеричному значению первого слова блока DPB быстро определять соответствующую директиву.

Пример использования восьмеричного значения первого слова блока DPB:

десятичное значение байтов:	5.	71.
восьмеричное значение байтов:	5	107
двоичное значение байтов:	101	01000 111
восьмеричное значение слова:	2507	
соответствующая директива —	SDAT $	

Таблица П.2

Восьмеричное значение первого слова блока DPB	Директива (макровызов)	Десятичное значение DIC	Длина блока DPB (в словах)
433	CMKT$	27.	1.
443	DECL$S	35.	1.
455	SPND$S	45.	1.
461	WSIG$S	49.	1.
463	EXIT$S	51.	1.
537	DSCP$S	95.	1.
541	ENCP$S	97.	1.
543	DSAR$S IHAR$S	99.	1.
545	ENAR$S	101.	1.
563	ASTX$S	115.	1.
575	GSSW$S	125.	1.
603	STOP$S	131.	1.
637	ULGF$S	159.	1.
1025	SRRA$	21.	2.
1035	EXST$	29.	2.
1037	CLEF$	31.	2.
1041	SETF$	33.	2.
1045	RDEF$	37.	2.
1047	RDAF$	39.	2.
1051	WTSE$	41.	2.
1065	EXIF$	53.	2.
1067	CRRG$	55.	2.
1071	ATRG$	57.	2.
1073	DTRG$	59.	2.
1075	GTIM$	61.	2.
1077	GTSK$	63.	2.
1121	RREF$	81.	2.
1153	SRDA$	107.	2.
1155	SPRA$	109.	2.
1157	SFPA$	111.	2.
1161	GMCX$	113.	2.
1165	CRAW$	117.	2.
1171	MAP$	121.	2.
1173	UMAP$	123.	2.
1207	STSE$	135.	2.
1227	ELVT$	151.	2.
1235	CRGF$	157.	2.
1237	ELGF$	159.	2.
1245	SPEA$	165.	2.
1247	SREA$	167.	2.
1255	SCAA$	173.	2.
1261	FEAT$	177.	2.
1311	MSDS$	201.	2.
1321	TFEA$	209.	2.
1325	RRST$	213.	2.
1405	GLUN$	5.	3.
1431	CSRQ$	25.	3.
1433	CMKT$	27.	3.
1447	RDXF$	39.	3.
1453	WTLO$	43.	3.
1457	RSUM$	47.	3.

Продолжение

Восьмеричное значение первого слова блока DPB	Директива (макровызов)	Десятичное значение DIC	Длина блока DPB (в словах)
1475	STIM$	61.	3.
1523	ABRT$	83.	3.
1531	EXTK$	89.	3.
1547	SVDB$	103.	3.
1551	SVTK$	105.	3.
1577	SNXC$	127.	3.
1605	USTP$	133.	3.
1611	STLO$	137.	3.
1617	CNCT$	143.	3.
1633	SCAL$S	155.	3.
1647	SREX$	167.	3.
1657	SWST$	175.	3.
1715	CPCR$	205.	3.
2007	ALUN$	7.	4.
2011	ALTP$	9.	4.
2101	GPRT$ GREG$	65.	4.
2113	RCVD$	75.	4.
2115	RCVX$	77.	4.
2213	RCST$	139.	4.
2223	EMST$	147.	4.
2313	MVTS$	203.	4.
2427	MRKT$	23.	5.
2505	SREF$	69.	5.
2507	SDAT$	71.	5.
2625	CRVT$	149.	5.
2655	SCLI$	173.	5.
2717	ACHN$	207.	5.
2717	DLON$	207.	5.
2717	SDIR$	207.	5.
3113	VRCD$	75.	6.
3115	VRCX$	77.	6.
3213	VRCS$	139.	6.
3317	GDIR$	207.	6.
3413	RQST$	11.	7.
3577	GCCI$	127.	7.
3601	CINT$	129.	7.
3615	SDRC$	141.	7.
3655	GCII$	173.	7.
3717	CLON$	207.	7.
3717	FSS$	207.	7.
4107	VSDA$	71.	8.
4215	VSRC$	141.	8.
4253	SMSG$	171.	8.
4615	SDRP$	141.	9.
5317	RLON$	207.	10.
5317	TLON$	207.	10.
5421	RUN$	17.	11.
6001	QIO$	1.	12.
6003	QIOW$	3.	12.
6413	SPWN$	11.	13.
6717	PFCS$	207.	13.
6717	PRMS$	207.	13.
7013	SPWN$	11.	14.
10013	PROI$	11.	16.
24577	GMCR$	127.	41.

3. Набор символов символьного кода КОИ-8

В табл. П.3 и П.4 приведен набор символов КОИ-8 в восьмеричном и шестнадцатеричном коде соответственно.

Таблица П.3

Восьмеричный код	Обозначение	Русское наименование	Восьмеричный код	Обозначение	Русское наименование
000	ПУС	Пусто	011	ГТ	Горизонтальная табуляция
001	НЗ	Начало заголовка			
002	НТ	Начало текста	012	ПС	Перевод строки
003	КТ	Конец текста	013	ВТ	Вертикальная табуляция
004	КП	Конец передачи			
005	КТМ	Кто там?	014	ПФ	Перевод формата
006	ДА	Подтверждение	015	ВК	Возврат каретки
007	ЗВ	Звонок	016	ВЫХ	Выход
010	ВШ	Возврат на шаг	017	ВХ	Вход
			020	АР1	Авторегистр 1

Продолжение

Восьмеричный код	Обозначение	Русское наименование	Восьмеричный код	Обозначение	Русское наименование
021	СУ1	Символ устройства 1	101	A	Прописные латинские буквы от A до Z
022	СУ2	Символ устройства 2			
023	СУ3	Символ устройства 3	102	B	
024	СТП	Стоп	103	C	
025	НЕТ	Отрицание	104	D	
026	СИН	Синхронизация	105	E	
027	КБ	Конец блока	106	F	
030	АН	Аннулирование	107	G	
031	КН	Конец носителя	110	H	
032	ЗМ	Замена	111	I	
033	АР2	Авторегистр 2	112	J	
034	РФ	Разделитель файлов	113	K	
035	РГ	Разделитель групп	114	L	
036	РЗ	Разделитель записи	115	M	
037	РЭ	Разделитель элементов	116	N	
			117	O	
040	SP	Пробел	120	P	
041	!	Восклицательный знак	121	Q	
042	"	Кавычки	122	R	
043	#	Номер	123	S	
044	$	Знак денежной единицы	124	T	
			125	U	
045	%	Процент	126	V	
046	&	Коммерческое «И»	127	W	
047	'	Апостроф	130	X	
050	(	Круглая скобка левая	131	Y	
051	)	Круглая скобка правая	132	Z	
			133	[	Квадратная скобка левая
052	*	Звездочка			
053	+	Плюс	134	\	Обратная косая черта
054	,	Запятая	135	]	Квадратная скобка правая
055	—	Минус			
056	.	Точка	136	^	Стрелка вверх
057	/	Косая черта	137	_	Подчеркивание
060	0	Цифры от 0 до 9	141	a	Строчные латинские буквы от a до z
061	1				
062	2		142	b	
063	3		143	c	
064	4		144	d	
065	5		145	e	
066	6		146	f	
067	7		147	g	
070	8		150	h	
071	9		151	i	
072	:	Двоеточие	152	j	
073	;	Точка с запятой	153	k	
074	<	Меньше	154	l	
075	=	Равно	155	m	
076	>	Больше	156	n	
077	?	Вопросительный знак	157	o	
100	@	Коммерческое «А»	160	p	
161	g		307	Ч	
162	r		310	Ш	
163	s		311	Щ	
164	t		312	Ъ	
165	u		313	Ы	
166	v		314	Ь	
167	w		315	Э	
170	x		316	Ю	
171	y		317	Я	
172	z		320	а	Строчные русские буквы от а до я
173	{	Фигурная скобка левая	321	б	
			322	в	
174	|	Вертикальная черта	323	г	
175	}	Фигурная скобка правая	324	д	
			325	е	
176	~	Волнистая черта	326	ж	
177	ЗБ	Забой	327	з	
241	Ё	Прописная русская буква Ё	330	и	
			331	й	
260	А	Прописные русские	332	к	
261	Б	буквы от А до Я	333	л	
262	В		334	м	

Продолжение

Восьмеричный код	Обозначение	Русское наименование	Восьмеричный код	Обозначение	Русское наименование
263	Г		335	н	
264	Д		336	о	
265	Е		337	п	
266	Ж		340	р	
267	З		341	с	
270	И		342	т	
271	Й		343	у	
272	К		344	ф	
273	Л		345	х	
274	М		346	ц	
275	Н		347	ч	
276	О		350	ш	
277	П		351	щ	
300	Р		352	ъ	
301	С		353	ы	
302	Т		354	ь	
303	У		355	э	
304	Ф		356	ю	
305	Х		357	я	
306	Ц		401	ё	Строчная русская буква ё

Таблица П.4

Младшая цифра	Старшая цифра 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	ПУС	АР1	SP	0	@	P	″	p				А	Р	а	р	
1	НЗ	СУ1	!	1	A	Q	a	q			Ё	Б	С	б	с	ё
2	НТ	СУ2	»	2	B	R	b	r				В	Т	в	т	
3	КТ	СУ3	#	3	C	S	c	s				Г	У	г	у	
4	КП	СТП	$	4	D	T	d	t				Д	Ф	д	ф	
5	КТМ	НЕТ	%	5	E	U	e	u				Е	Х	е	х	
6	ДА	СИН	&	6	F	V	f	v				Ж	Ц	ж	ц	
7	ЗБ	КБ	′	7	G	W	g	w				З	Ч	з	ч	
8	ВШ	АН	(	8	H	X	h	x				И	Ш	и	ш	
9	ГТ	КН	)	9	I	Y	i	y				Й	Щ	й	щ	
A	ПС	ЗМ	*	:	J	Z	j	z				К	Ъ	к	ъ	
B	ВТ	АР2	+	;	K	[	k	{				Л	Ы	л	ы	
C	ПФ	РФ	,	<	L	\	l	|				М	Ь	м	ь	
D	ВК	РГ	—	=	M	]	m	}				Н	Э	н	э	
E	ВЫХ	РЗ	.	>	N	^	n	~				О	Ю	о	ю	
F	ВХ	РЭ	/	?	O	—	o	ЗБ				П	Я	п	я	

4. Отличие СУД ОС РВМ от СУД-2 ОСРВ

В систему управления данными ОСРВМ введены следующие изменения:

добавлены новые макрокоманды операций над каталогами $ENTER, $PARSE, $REMOVE, $RENAME и $SEARCH и связанные с ними новые поля FNB, RSA, RSZ и RSS управляющего блока NAM;

обладает новыми возможностями при использовании символов групповой операции в спецификации файла;

расширены возможности совместного доступа и связанные с ним маски FB$UPI и FB$NIL для поля SHR управляющего блока FAB;

предусмотрен прямой доступ по ключу к последовательному файлу с записями фиксированной длины (аналогичный прямому доступу по ключу к относительному файлу);

возможна обработка выходных записей формата печати и связанный с ним символ FB$PRN поля RAT блока FAB;

предусмотрен новый последовательный доступ к блокам. Доступ к блокам, ранее называемый вводом-выводом блоков, теперь называется VBN-доступом (доступ по виртуальному номеру блока);

для копирования СУД-файлов может использоваться доступ к блокам без изменений атрибутов файлов;

введены дополнительные возможности подпрограммы успешного завершения для макрокоманд операций над файлами $ CLOSE, $CREATE, $ DISPLAY, $ ERASE, $ EXTEND и $OPEN);

поддерживается вывод записей формата потока и VFC-записей на нефайловые устройства, вывод относительных и индексных файлов на нефайловые устройства, где они будут рассматриваться как последовательные;

макрокоманды инициализации СУД $ INIT и $INITIF не описываются. Их прежние функции в СУД ОСРВМ не требуются, так как СУД является самоинициализируемой системой. Однако программы, использующие макрокоманды $INIT и SINITIF, выполняются в СУД ОСРВМ;

каждый блок XAB теперь имеет индивидуальное имя. Новыми именами являются:

блок ALL — размещение области;
блок DAT — дата файла;
блок KEY — ключи файла;
блок PRO — защита файла;
блок SUM — сводка о файле;

введены следующие макрокоманды объявления символов:

FAB $ BT — значение и символы масок блока FAB;
NAM $ BT — значение и символы масок блока NAM;
RAB $ BT — значение и символы масок блока RAB;
XAB $ BT — значение и символы масок блока XAB;
XBAOF$ — символы блока ALL;
XBDOF$ — символы блока DAT;
XBKOF$ — символы блока KEY;
XBPOF$ — символы блока PRO;
XBSOF$ — символы блока SUM;

управляющий блок FAB имеет новое поле LRL (длина максимальной записи) для последовательных файлов;

изменен размер управляющего блока DAT (даты и времени) от 36 до 46 восьмеричных бит;

для магнитной ленты в формате ЕС ЭВМ ОСРВМ поддерживает записи фиксированной длины (короче 18 байт);

управляющие символы CTRL/Z и ESC не распознаются в качестве ограничителя записи файлов формата потока.

СУД ОСРВМ заполняет наивысший блок файла формата потока нулевыми символами до физического конца (не только до конца текущего блока). Это означает, что файлы формата потока, созданные СУД, могут быть прочитаны программами, которые не распознают значение конца файла (EOF) из заголовка файла.

ЛИТЕРАТУРА

1. Операционная система ОСРВ СМ ЭВМ/ Г.А. Егоров, В.Л. Кароль, И.С. Мостов и др. — М.: Финансы и статистика, 1987. — 271 с.

2. СМ ЭВМ: комплексирование и применение/Под ред. Н.Л. Прохорова. — 2-е изд., перераб. и доп. — М.: Финансы и статистика, 1987. — 304 с.

3. Столяр Л.Н., Шапошников В.А. Средства проверки работоспособности оборудования СМ ЭВМ. — М.: Финансы и статистика, 1986.— 142 с.