БИБЛИОТЕЧКА
ПРОГРАММИСТА

В. И. ЗУЕВ, В. М. КРЮКОВ, В. И. ЛЕГОНЬКОВ

УПРАВЛЕНИЕ ДАННЫМИ В ВЫЧИСЛИТЕЛЬНОМ ЭКСПЕРИМЕНТЕ

МОСКВА «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
1986

ББК 22.18
393
УДК 519.6

Зуев В. И., Крюков В. М., Легоньков В. И. Управление данными в вычислительном эксперименте.— М.: Наука. Гл. ред. физ.-мат. лит., 1986.— 160 с.

Управление данными в вычислительном эксперименте — один из основных методов исследования и проектирования сложных физико-технических объектов с помощью ЭВМ. Рассмотрены отечественные и зарубежные системы управления данными с точки зрения требований, возникающих при проведении вычислительного эксперимента. Изложен подход к решению проблемы, на основе которого создана система управления данными для многомашинного вычислительного комплекса ЭВМ БЭСМ-6. Проведен анализ этой системы и приведены ее показатели эффективности и надежности, удовлетворяющие сформулированным в книге требованиям.

Для специалистов по обработке данных в вычислительных системах.

Табл. 2. Ил. 73. Библиогр. 112 назв.

Рецензент кандидат физико-математических наук
В. Р. Хисамутдинов

З $\frac{1702070000-104}{053(02)-86}$ 30-86

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

В книге рассматриваются проблемы управления данными в вычислительном эксперименте — одном из основных методов исследования и проектирования сложных физико-технических объектов (энергетических установок, летательных аппаратов, строительных сооружений и др.) с помощью ЭВМ.

Вычислительный эксперимент включает в себя совокупность большого количества взаимосвязанных расчетов по различным программам, а каждый расчет — переработку информации большого объема.

Излагается подход к созданию систем управления данными, базирующийся на исследовании процессов движения информации между человеком и вычислительной системой при проведении расчета и от расчета к расчету внутри вычислительной системы. Описывается модель управления совокупностью большого количества потоков данных в неоднородной внешней памяти ЭВМ, на основе анализа которой разработаны архитектура и функциональная структура системы управления данными для многомашинного вычислительного комплекса ЭВМ БЭСМ-6. Показывается эффективность функционирования этой системы как в смысле обеспечения потребностей человека, так и в смысле использования ресурсов процессора и памяти.

Большое внимание уделяется проблеме надежности управления данными. Это вызвано высокой ответственностью получаемых в расчетах данных, обусловленной экономическими причинами и необходимостью выбора оптимальных и безопасных вариантов создаваемых объектов. Описывается подход к решению проблемы надежности управления данными в сложной операционной обстановке, определяемой технологией вычислительного эксперимента, конфигурацией оборудования и операционной системой ЭВМ, а также наличием разнообразных внутренних и внешних возмущений.

Рассматриваются два основных аспекта этой проблемы: защита данных от возмущений и восстановление данных при их искажении (уничтожении) в результате возмущений. С этой целью

исследуются модели систем управления данными. В результате определяются факторы, влияющие на надежность систем, уязвимые процессы и компоненты систем. Описываются реализация выбранного подхода, алгоритмы и мероприятия, обеспечивающие достаточно высокую надежность управления данными.

Рассматриваются вопросы, связанные с разработкой и эксплуатацией конкретной системы управления данными (УПД ДИАПАК). Приводятся анализ работы этой системы и ее технико-экономические характеристики, полученные в процессе промышленной эксплуатации.

Книга может быть полезна широкому кругу специалистов по обработке данных в вычислительных системах.

ВВЕДЕНИЕ

Возрастание роли вычислительного эксперимента при проведении научных исследований и создании образцов новой техники [1—6] и связанное с этим усложнение математических методов, увеличение количества проводимых расчетов, повышение требований к точности и сокращению сроков проведения расчетов, создание многомашинных вычислительных комплексов [7—13], внедрение диалоговых форм взаимодействия пользователя с ЭВМ [14—17] привели к резкому увеличению объема данных в информационных базах вычислительных центров. В этих условиях большое значение и актуальность приобретает проблема автоматизации функций информационного обеспечения расчетов при решении разнообразных задач науки и техники.

Пути решения этих вопросов связаны с повышением уровня управления информацией, хранимой и обрабатываемой в вычислительной системе (ВС). Одним из таких путей является создание систем управления данными (СУД), основанных на наиболее эффективных способах организации, идентификации, классификации, запоминания и выборки данных, интеграция на основе СУД информационных баз вычислительных центров (ВЦ), внедрение единой технологии организации хранения данных и обращения к ним [19—47].

Постоянно возрастающая сложность научно-технических расчетов, связанная с возрастанием сложности решаемых задач, требует развития структуры вычислительных средств, оснащения их новыми устройствами внешней памяти с различными возможностями поиска и выборки данных, организации новых форм взаимодействия пользователя с ВС, совершенствования технологии хранения данных и обращения к ним. Все это в совокупности предъявляет к системам управления данными ряд требований, без учета которых немыслимо удовлетворительное решение указанной выше проблемы.

Усложнение решаемых задач и, как следствие этого, увеличение объема и усложнение структур обрабатываемых данных,

массовость решения этих задач и информационная взаимосвязь между ними требуют:

— организации длительного хранения данных больших объемов и разнообразных структур;

— предоставления средств классификации и поиска данных по различным признакам;

— обеспечения одновременного использования общих данных несколькими задачами;

— различного представления данных в прикладной программе и во внешней памяти ЭВМ;

— унификации способов доступа к данным;

— простоты общения с системой при передаче данных на хранение или обращении к данным.

Создание многомашинных вычислительных комплексов (МВК) с общим полем внешней памяти и общей терминальной сетью приводит к необходимости:

— обеспечения управления общими ресурсами памяти комплекса;

— организации единой информационной базы комплекса, когда данные, полученные на одной ЭВМ, становятся одновременно доступными и для всех остальных ЭВМ.

Оснащенность ЭВМ неоднородными по составу устройствами внешней памяти предъявляет требования:

— обеспечения независимости прикладных программ от типов внешних запоминающих устройств;

— обеспечения эффективной загрузки оборудования с учетом технических возможностей каждого устройства;

— рационального размещения информационных массивов по физическим типам памяти.

Использование ВС в режиме разделения времени и диалого-пакетная форма эксплуатации предъявляют требования:

— высокой скорости доступа к данным;

— простоты языка общения с системой управления данными;

— эффективной реализации процедур доступа к данным;

— обеспечения сохранности и защиты данных.

В масштабах ВЦ встает задача автоматизации ряда звеньев в технологическом цикле подготовки и решения производственных задач, т. е. возникает необходимость:

— создания единой технологии хранения данных и обращения к ним;

— автоматизации рутинной работы по учету ресурса внешней памяти и размещаемых в ней данных;

— оперативного предоставления пользователям справочной информации о наличии данных, о свободном и использованном ресурсе памяти и т. п.

Перечисленные требования находят определенные решения в различных системах управления данными, разрабатываемых и работающих как в нашей стране, так и за рубежом.

Являясь посредником между прикладными программами и данными, хранимыми во внешней памяти ЭВМ, СУД разделяют логический и физический уровни представления данных. Такое разделение на ЭВМ третьего поколения реализовано во многих системах [20—22, 27—34].

Примером подобной реализации СУД на комплексе двух вычислительных машин IBM/360-168 может служить работа [35]. Система управления данными на этом комплексе ЭВМ выполняет размещение массивов в общедоступной памяти большой емкости. Преимущества централизованного хранения данных состоят в относительной легкости и простоте поддержания информационной базы в непротиворечивом состоянии, экономии памяти, удобстве для пользователя и др.

При организации централизованного хранения данных возникает задача управления общими информационными ресурсами. Главная трудность в управлении данными на комплексе ЭВМ заключается в упорядочении доступа к файлам и исключении конфликтных ситуаций при одновременном обращении к файлу из нескольких задач, т. е. в тех случаях, когда две или более задач одновременно модифицируют один и тот же файл.

Методы, реализующие распределенный доступ к файлам в комплексах ЭВМ, основаны на аппаратной и/или программной блокировке доступа к файлу [9, 12, 48]. В случае возникновения аварийной ситуации предусматриваются специальные средства, обеспечивающие автоматический выход системы из состояния блокировки, не препятствующие при этом одновременному доступу к файлам из нескольких ЭВМ.

Вопросам рационального размещения данных во внешней памяти ЭВМ в последние годы уделяется особое внимание. Поиск рационального размещения сводится в основном к решению двух задач: оптимизации размещения массивов на отдельных томах внешней памяти (магнитных лентах и магнитных дисках) [42—54] и выбору оптимального распределения информации по физическим уровням памяти ЭВМ [36—39, 55—60]. Большинство работ посвящено решению первой задачи. В частности, наиболее полно разработан вопрос определения порядка размещения массивов на магнитной ленте (МЛ).

Задача оптимального размещения информации во внешней памяти, состоящей из нескольких физических уровней, исследована в меньшей степени. Представляет интерес попытка решить эту задачу путем разработки иерархических систем памяти, которые обеспечивают размещение наиболее часто используемых

данных в более быстрой памяти ЭВМ. Возникающие при этом встречные перемещения данных с одного уровня на другой реализуются с помощью алгоритмов замещения [37—40, 57—60].

Двухуровневая система внешней памяти реализована в УПД-6 [30], одной из наиболее известных в СССР систем управления данными для БЭСМ-6. Для общего пользования создается локальный банк данных (ЛБД) [59—60]. Тома внешней памяти, выделенные под ЛБД, разбиваются на две группы. Одна из них состоит из магнитных дисков (МД), постоянно установленных на устройствах, и образует основное пространство ЛБД. Другая группа томов, состоящая из МД и МЛ, образует вторичное пространство. Данные, находящиеся на хранении в ЛБД, могут быть легко перемещены с одного носителя на другой. Наиболее часто используемые данные размещаются в основном пространстве. Данные, к которым обращения редки, размещаются во вторичном пространстве.

Перемещения данных из основного пространства во вторичное пространство инициирует администратор данных, который выполняет функции сопровождения ЛБД. Согласно выбранному в ЛБД алгоритму замещения из основного пространства «выталкиваются» те данные, к которым дольше всего не было обращения. Если пользователю потребовались данные, находящиеся во вторичном пространстве, то он может дать заявку администратору на «вталкивание» данных или выполнить перепись данных в основное пространство сам, используя специальную утилиту.

В кратком обзоре иерархических систем памяти, приведенном в [39], отмечаются две операционные системы MALTICS и TSS/360, обеспечивающие размещение файлов на различных уровнях внешней памяти. Процедура передачи данных на магнитные ленты автоматизирована и выполняется ежедневно. Пользователю также предоставляется набор команд, дающих возможность определить, какие из его файлов перемещены во вторичную память, и позволяющих извлечь файлы из вторичной памяти.

В работе [36] описана программная система, обеспечивающая автоматическое распределение непосредственно доступной памяти на магнитных дисках, выполняющих роль оперативной области, и пересылку на вторичную память (в основном на МЛ) тех массивов данных, к которым более двух месяцев не происходило обращение.

Вопросам рационального размещения данных во внешней памяти ЭВМ большое внимание уделяется при разработках систем управления базами данных (СУБД), которые в настоящее время получают широкое распространение как в СССР, так и за рубежом. С целью улучшения эксплуатационных характеристик в СУБД предусматривается возможность изменения организации хранения данных. Эта возможность реализуется путем реконст-

рукции баз данных, за проведение которой отвечает администратор данных [37]. При этом в некоторых системах допускается перемещение части данных из медленной памяти в более быструю и наоборот. Возможность такого перемещения данных в соответствии со степенью активности их использования реализована, например, в системах СИНБАД-2 [43], ОКА [44], IMS [41, 42].

К числу работ, посвященных решению технологических проблем, относятся, в частности, [45, 46]. В вычислительном центре Лёйна [45] обслуживание всего фонда магнитных лент производит система, ядром которой служит программа, использующая систему банка данных DBS/R [47]. Функции сопровождения архива магнитных лент выполняет администратор. Он сообщает системе сведения о порядке использования МЛ, о сроках хранения данных на лентах, о порядке уничтожения данных и т. п. Для записи результатов прикладных задач устанавливается очередная свободная лента, рекомендованная системой. Сервисные средства системы позволяют администратору в удобной форме выполнять формирование архива МЛ, внесение в него изменений, стирание массивов данных. Система имеет язык опроса, средства использования терминальных устройств и средства извлечения данных.

В [46] предлагается набор прикладных программ, позволяющих решить задачу управления большой библиотекой томов внешней памяти. Совокупность сведений о всех томах образует логический архив. Программы реализуют создание логического архива, его модификацию, выборку и накопление сведений о частоте обращений к томам, об использовании их, выдают справочную информацию. Для получения интегрированных результатов о работе многомашинного комплекса предлагается объединять сведения, накопленные на отдельных ЭВМ, и совместно обрабатывать их.

В этих системах сделана попытка полностью снять с пользователей заботу о распределении внешней памяти и упростить технологию работы с ней. Автоматическое ведение архива дает ряд преимуществ: автоматизируется рабочий процесс, появляется возможность оперативного получения справки об общем состоянии фонда томов с различных точек зрения, упрощаются функции ведения архива (инвентаризация, учет свободных и занятых ресурсов и т. п.).

В связи со значительным влиянием технологии обработки данных на производительность труда пользователей и эффективность функционирования ВС в настоящее время возникает необходимость создания таких СУД, которые способны обеспечить более мощную программную поддержку той технологии обработки данных, которая характерна для конкретного ВЦ. Разработка

таких систем приобретает важное значение для многих ВЦ, обслуживающих широкий круг пользователей в пакетном и диалоговом режимах, в условиях массового счета задач различной проблематики и возникающих при этом больших объемов данных.

В данной книге рассматривается круг вопросов, связанных с исследованием методов управления данными при проведении сложных физических расчетов и разработкой соответствующей системы управления данными. При этом учитываются свойства решаемых задач, технология проведения вычислительного эксперимента, архитектура многомашинного вычислительного комплекса. Главная цель состоит в повышении производительности вычислительных средств и труда пользователей (программистов, вычислителей, операторов, администраторов данных), предоставлении возможностей для решения более сложных задач информационного обслуживания, в том числе в повышении эффективности использования неоднородной по составу внешней памяти и внедрении прогрессивных методов хранения данных в памяти ЭВМ.

Достижение указанной цели осуществляется путем построения наиболее подходящей модели управления данными и ее реализации в виде системы для многомашинного вычислительного комплекса. При этом решаются следующие задачи:

— логической организации информационной базы ВЦ;

— организации двухуровневой системы внешней памяти;

— автоматического распределения и перераспределения памяти и размещения в ней данных;

— организации совместного использования общих данных.

В книге рассматриваются также вопросы выбора функциональной структуры системы управления данными и модели функционирования ее, разработки и реализации конкретных модулей — компонент системы, включения их в состав имеющегося математического обеспечения, внедрения системы в промышленную эксплуатацию на реальном многомашинном комплексе и обеспечения новой технологии управления информацией в вычислительном центре.

Предложенный подход к созданию систем управления данными, базирующийся на выборе модели функционирования системы и модели управления данными с учетом особенностей задач и технологии решения их в ВЦ, может быть использован при создании систем управления данными с другой проблемной ориентацией. Предложенные модель управления данными и принципы создания единой информационной базы многомашинного вычислительного комплекса могут быть использованы при создании систем управления данными для неоднородных комплексов ЭВМ с применением более совершенных средств запоминания, хранения и поиска информации.

Результаты исследований использованы при разработке системы управления данными УПД ДИАПАК [61—68] для МВК БЭСМ-6 [8—10], работающем под управлением операционной системы (ОС) ДИАПАК [11].

Внедрение УПД ДИАПАК в промышленную эксплуатацию позволило:

— упростить технологию проведения вычислительного эксперимента на этапах программирования, отладки программ, подготовки задач к решению, проведения расчетов на ЭВМ;

— создать единую информационную базу многомашинного комплекса;

— унифицировать работу с данными, находящимися на длительном хранении;

— иметь одновременный доступ к общим данным;

— автоматизировать функции учета данных;

— повысить готовность наиболее активных данных;

— более эффективно использовать общий ресурс внешней памяти, состоящей из накопителей с различными техническими характеристиками;

— повысить эффективность использования дисковой памяти;

— сократить нагрузку на ленточную память;

— осуществить централизованный контроль за использованием внешней памяти.

Другой круг вопросов рассмотренных в книге, связан с проблемой надежности систем управления данными. Надежность СУД во многом определяет достоверность и сохранность данных, что в свою очередь является решающим условием эффективной работы ВС, так как при скоплении в памяти ЭВМ большого количества файлов повышается их уязвимость — возможность потери или искажения информации в результате сбоев и отказов системы. Такие последствия сбоев и отказов способны резко снизить (а в ряде случаев свести к нулю) эффект от внедрения и эксплуатации системы, а также привести к существенному снижению производительности ВС.

Основными причинами ненадежности систем управления данными является объем и сложность этих систем (по структуре, связям и функциям), вычислительная трудоемкость реализованных в них алгоритмов обработки различных информационных структур и централизованное хранение данных. Сбои и отказы таких систем обусловлены не только их собственными внутренними ошибками, но и возмущениями окружающей среды. Как показывает практика, и те, и другие неизбежны, что характерно для больших и сложных программных систем [69—72]. Этими обстоятельствами, а также появлением аппаратуры вычислительных средств, обладающей высокой производительностью и достаточ-

ной надежностью, обусловлены исследования, связанные с повышением надежности программного обеспечения, развернувшиеся в 70-е годы в нашей стране и за рубежом.

Тематику этих исследований составили следующие вопросы [69—76]:

— формулирование понятий, связанных с надежностью программного обеспечения;

— определение факторов, влияющих на надежность;

— выбор и обоснование критериев надежности;

— определение требований к программному обеспечению;

— разработка методов проектирования и производства программных систем, проверки их правильности, тестирования и отладки;

— моделирование и анализ надежности систем;

— управление разработками и подготовка специалистов.

Результаты этих исследований, опубликованные в отечественной и зарубежной литературе за последние 5 лет, свидетельствуют о том, что не все проблемы удалось решить [70—74]. Кроме того, все «хорошие идеи», предлагаемые в литературе, не всегда можно проверить на практике, так как эксперименты по разработке программного обеспечения очень дороги.

Тем не менее эти исследования стимулировали развитие автоматизированных систем проверки, отладки и оценки программного обеспечения, систем автоматического конструирования программ, специализированных мониторов для тестирования программ и т. п., которые в перспективе позволят справиться со сложностью программных систем, свести ее к минимуму.

Определяемый этими исследованиями подход ставит своей целью предупреждение появления ошибок при изготовлении программ. Он является достаточно привлекательным, поскольку на создаваемые программы не возлагается дополнительных функций. Однако, очевидно, этот подход не может гарантировать полное отсутствие ошибок [72, 74].

Другой подход к решению проблем надежности, заключающийся в обеспечении устойчивости систем к внешним и внутренним возмущениям посредством введения различной избыточности для обнаружения возмущений и восстановления работоспособного состояния, применяется при разработке сложных технических систем (в частности, ЭВМ) [70—73, 77, 78]. Первые попытки адаптации и применения его в области программирования были сделаны при создании программного обеспечения для систем реального времени: бортовых систем управления, систем управления воздушным движением, автоматизированных систем управления ответственными отраслями промышленности [71, 79, 80].

Недостатком этого подхода является наличие затрат на разработку дополнительных функций контроля, а также процессорного времени и памяти на выполнение этих функций. Высокий уровень ответственности расчетов, проводимых в процессе вычислительного эксперимента, естественно определяет и высокие требования к обеспечению достоверности полученных данных, что в свою очередь предъявляет соответствующие требования к надежности систем управления данными. Обеспечить приемлемое выполнение этих требований сегодня возможно путем объединения рассмотренных выше подходов. Традиционные приемы и методы первого подхода позволяют сократить количество ошибок в процессе создания системы. Методы второго подхода позволяют дополнить систему средствами, обеспечивающими устойчивое функционирование при наличии ошибок (внутренних возмущений) и влияния окружающей среды (внешних возмущений). Такое объединение подходов может быть реализовано следующим образом.

При разработке архитектуры системы используются традиционные приемы и методы:

— структуризации программных компонент и управляющей информации, необходимой для функционирования системы;

— стандартизации интерфейсов между программными компонентами, самой системой и ОС;

— унификации представления программных и информационных компонент системы;

— упорядочения связей между программными компонентами системы.

Этим упрощается система, облегчается и ускоряется ее изготовление, уменьшается время на отыскание и устранение программных ошибок и искажений управляющей информации.

При проектировании взаимодействия пользователя с системой основное внимание уделяется достижению простоты формы этого взаимодействия и его соответствия навыкам работы пользователя с данными. Этим преследуются две цели: минимизация ошибок пользователя и исключение их влияния на работу системы.

После разработки структуры системы проводится анализ с целью определения источников возмущений и выявления уязвимых компонент системы (чувствительных к возмущениям), а также анализ влияния искажений этих компонент на функционирование системы в целом и сохранность файлов. Методика такого анализа может основываться на результатах исследования известных систем управления данными, а также на опыте разработки и эксплуатации крупных систем общего назначения.

По результатам анализа определяются методы обеспечения устойчивости системы, основанные на использовании информационной и программной избыточностей. При этом решаются три основные задачи: ограничение аварийных последствий возмущений (таких, как прекращение функционирования системы, искажение жизненно важных объектов управляющей информации, искажение файлов и т. п.), контроль состояния системы, восстановление управляющей информации, файлов и системы.

Таким образом, методы обеспечения устойчивости направлены:

— на обнаружение возмущений, их диагностику, ограничение или ликвидацию их последствий;

— на поддержание сохранности управляющей информации и файлов;

— на оперативный и автономный контроль состояния управляющей информации, структуры файлов и системы в целом;

— на оперативное и автономное восстановление управляющей информации, файлов и функционирования системы.

Так закладывается надежность системы на стадии ее проектирования.

В процессе изготовления системы запроектированная надежность обеспечивается:

— соблюдением технологии структурного и модульного программирования;

— проверкой логики работы каждого модуля;

— тестированием путей в модуле;

— созданием или имитацией аварийных ситуаций для определения фактической устойчивости алгоритма, модуля и системы в целом;

— проверкой работоспособности системы в условиях МВК.

При эксплуатации системы должна поддерживаться достигнутая надежность. Для этого необходимы:

— подготовка персонала, отвечающего за нормальное функционирование системы, и средства программной поддержки его деятельности;

— точные инструкции по эксплуатации системы, ее восстановлению после отказов, а также восстановлению управляющей информации и файлов после их искажения или потери;

— сбор статистической информации о всех обнаруженных ошибках, возникших сбоях или отказах в работе системы, их причинах, последствиях и времени устранения, анализ этой информации и принятие соответствующих мер.

Недостаточная корректность системы на первых этапах эксплуатации должна компенсироваться способностью противостоять собственным ошибкам, исключать или ограничивать их отрица-

тельные последствия для сохранности файлов. Это достигается, в частности, организацией:

— встроенного контроля работы модулей системы, что облегчает поиск трудноуловимых ошибок;

— автоматизированного контроля состояния управляющей информации, ее дублирования и в ряде случаев восстановления;

— оповещения оператора о всех обнаруженных отклонениях, принятых и необходимых мерах, что позволяет сократить время простоя системы при возникновении отказа;

— защиты жизненно важных компонент системы от случайного или преднамеренного искажения со стороны пользователя;

— контроля действий обслуживающего персонала над этими компонентами и регистрации выполняемых им коррекций.

Следует подчеркнуть, что повышение надежности системы управления данными связано с увеличением затрат на ее создание и возрастанием издержек при ее функционировании (увеличением времени выполнения операций, расхода оперативной памяти и нагрузки на внешнюю память). Все это снижает производительность системы. Поэтому естественно требование при создании системы управления данными, которое состоит в минимизации затрат и издержек при одновременном обеспечении приемлемого (высокого) уровня надежности.

В книге рассмотрен круг вопросов, связанных с исследованием и обеспечением надежности управления данными.

Формируется подход к решению проблем надежности с учетом требований высокой достоверности данных, используемых при проведении ответственных расчетов, сложности операционной обстановки, образуемой конфигурацией оборудования и ОС, и различных внешних и внутренних возмущений.

Рассматриваются модели систем управления данными с точки зрения проблем надежности. На основе анализа этих моделей определяются факторы, влияющие на надежность систем, уязвимые процессы и компоненты систем.

Описываются методика для реализации выбранного подхода, алгоритмы и мероприятия по обеспечению надежности управления данными в сложных физических расчетах. Эта методика опробована при создании системы управления данными УПД ДИАПАК. В рамках системы реализованы рассмотренные выше приемы и методы предупреждения ошибок и обеспечения ее устойчивого функционирования.

ГЛАВА 1

ИССЛЕДОВАНИЕ ПРОЦЕССОВ УПРАВЛЕНИЯ ДАННЫМИ

1.1. Анализ информационной базы

Вычислительный эксперимент, о котором говорилось выше, состоит из следующих шагов [4]:

— выбор физического приближения и формулировка математической модели;

— аппроксимация математической модели и разработка вычислительного алгоритма;

— создание программы, реализующей вычислительный алгоритм;

— проведение расчетов на ЭВМ и получение результатов;

— анализ результатов, сравнение с физическим экспериментом, уточнение модели;

— если необходимо, повторение перечисленных шагов до тех пор, пока не будет получен удовлетворительный результат.

Современный этап развития вычислительного эксперимента характеризуется наличием двух взаимосвязанных тенденций [5]:

— усложнением отдельных задач, что приводит к созданию множества программ, каждая из которых соответствует какой-то части задачи, а их объединение представляет всю задачу;

— комплексным, взаимосвязанным решением задач, что приводит к созданию комплексов и сетей программ.

Таким образом, задачам, решаемым при проведении вычислительного эксперимента, присущи такие свойства, как

— массовость, которая определяется большим числом расчетов;

— частая изменяемость, которая следует из постоянного уточнения модели;

— связанность, которая обусловлена разбиением сложных задач на части.

Перечислим основные группы данных, составляющих информационную базу, охарактеризуем их назначение и использование (рис. 1.1).

Прежде всего отметим наличие библиотек программ и библиотек данных, описывающих свойства исследуемых объектов. Развитие знаний об объекте исследования и уточнение модели приводят к появлению новых версий этих библиотек. Совокупность всех версий библиотек отражает историю развития модели, а длительное хранение их дает возможность при необходимости возвратиться назад без дополнительных затрат и изменить направление поиска.

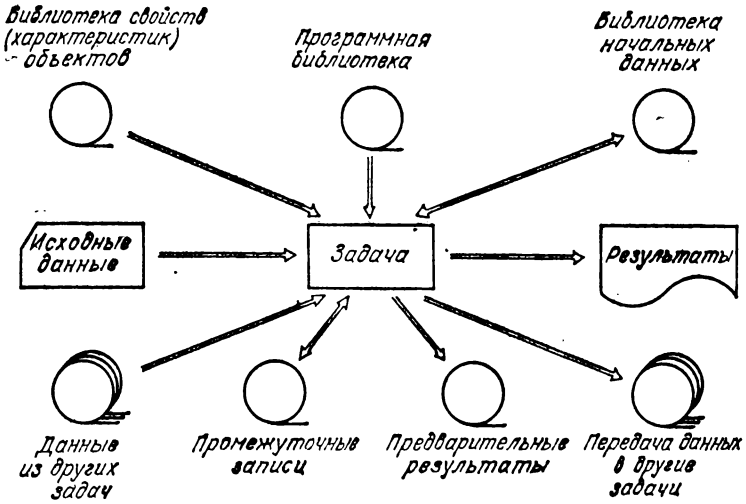


Рис. 1.1. Основные информационные массивы задачи

При проведении большого числа расчетов создаются библиотеки начальных данных рассчитываемых вариантов задач. Они используются как для повторения некоторых расчетов по более совершенным программам, так и для формирования начальных данных других вариантов задач путем замены значений лишь части имеющихся в библиотеке начальных данных.

Длительное время решения задачи и возможность сбоя вычислительной системы или возникновения аварийной ситуации в задаче вызывают необходимость фиксации и сохранения состояния задачи через определенные промежутки времени. Эти состояния образуют «след» решения задачи и позволяют в случае необходимости продолжить счет, начиная с некоторого промежуточного состояния, или вернуться к некоторому моменту времени и выполнить перерасчет.

В ходе решения задачи по некоторым «событиям» снимается предварительная результатная информация и накапливается

в виде результатного массива. Этот массив может быть неоднократно использован для обработки и выдачи результатов на устройства ЭВМ (алфавитно-цифровую печать, графопостроитель и т. п.).

Информационная связь между задачами, заключающаяся в передаче множеств данных из одной задачи в другую, требует накопления и хранения этих данных до тех пор, пока они не будут использованы.

Проведение массовых расчетов влечет появление множеств библиотек начальных данных, промежуточных, результатных и передаваемых массивов, а разнообразие исследуемых проблем вызывает появление множеств библиотек программ и библиотек характеристик исследуемых объектов.

Перечисленным множествам массивов данных присуще одно общее свойство: каждое множество вызывает в вычислительной системе свой информационный поток.

Пусть $M = \{m_i\}$ — множество массивов данных некоторого назначения. Каждый элемент этого множества, т. е. массив m_i, появляется в некоторый момент времени t_i и занимает объем памяти $v_i \neq 0$.

Определим функцию $V(t)$, характеризующую изменение объема данных в множестве M в зависимости от времени:

$$V(t) = \begin{cases} u_k & \text{в точках } t_k, \\ u_k + \dfrac{(u_{k+1} - u_k)(t - t_k)}{(t_{k+1} - t_k)} & \text{на интервалах } (t_k, t_{k+1}), \end{cases}$$

где $u_k = \sum_{i=1}^{k} v_i$.

Легко видеть, что функция $V(t)$ монотонно возрастает. Поскольку в реальных условиях памяти неограниченного объема не существует, то для множества M выделяется ресурс памяти W (лимит), превзойти который функция $V(t)$ не может. Это приводит к тому, что создание очередного массива в множестве M будет сопровождаться уничтожением одного или нескольких массивов этого множества, т. е. множество M будет содержать ограниченное число элементов.

Время, в течение которого массив существует в множестве, назовем *сроком хранения массива* ($T_{\text{хр}}$). Срок хранения зависит от скорости поступления данных $S(t) = V(t)/t$, объема памяти W и условия необходимости хранения данных.

Массив m_i может находиться либо в состоянии использования одной или несколькими задачами (активное состояние), либо в состоянии простоя (пассивное состояние).

На интервале (t_i, ∞) определим функцию

$$a_i(\tau, t) = \begin{cases} 1, & \text{если существует момент времени } t_0 \in [t-\tau, t], \\ & \text{когда массив } m_i \text{ находится в активном состоянии,} \\ 0, & \text{если в любой момент времени } t_0 \in [t-\tau, t] \text{ массив} \\ & m_i \text{ находится в пассивном состоянии.} \end{cases}$$

Эта функция характеризует использование массива m_i во времени. Поведение функции зависит от значения параметра τ. Если $\tau = 0$, то каждый переход массива из одного состояния в другое будет отмечен на графике скачком (рис. 1.2). С увеличением параметра τ количество скачков N будет уменьшаться, а функция $a_i(\tau, t)$ будет принимать значение 1 в большем числе точек области определения.

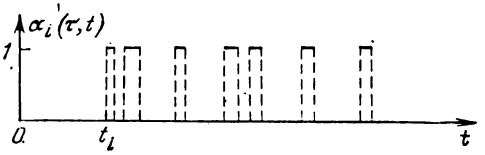


Рис. 1.2. График функции $a_i(\tau, t)$ при $\tau = 0$

Обозначим $\tau_i = \inf_{N=1} \tau$. Такое значение всегда существует, так как массив m_i со временем устаревает и подлежит уничтожению. Момент перехода функции $a_i(\tau, t)$ из 1 в 0 обозначим t_i' (рис. 1.3).

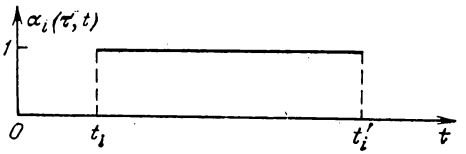


Рис. 1.3. График функции $a_i(\tau, t)$ при $\tau = \tau_i$

Время $t_i' - t_i$ назовем *активным периодом массива* и обозначим $T_{\text{акт}}$. Активный период массива всегда меньше его срока хранения, а в некоторых случаях разница между ними очень велика, т. е. $T_{\text{акт}} \ll T_{\text{хр}}$. Это можно объяснить как субъективным желанием пользователей дольше хранить «свои» данные, так и объективной производственной необходимостью длительного хранения тех или иных данных и т. п. Такие факторы, как большие объемы данных, высокая интенсивность их использования в период $T_{\text{акт}}$, существенная разница во временах $T_{\text{хр}}$ и $T_{\text{акт}}$ при наличии неоднородной по составу внешней памяти, приводят к не-

обходимости создания системы хранения, в которой данные для обработки в течение периода $T_{акт}$ размещаются в «быстрой» памяти, а для длительного хранения в течение периода $T_{хр} - T_{акт}$ размещаются в «медленной» памяти.

1.2. Модель управления данными

Построим модель управления данными в двухуровневой памяти и исследуем ее с целью выбора организации внешней памяти, наилучшим образом учитывающей характеристики различных информационных потоков. На этой основе сформулируем затем требования к системе управления данными.

Модель управления данными в двухуровневой памяти может быть представлена в виде ориентированного графа (рис. 1.4). Вершины x_1, x_2 — уровни памяти (x_1 — верхний уровень, x_2 — нижний уровень), петля a свидетельствует о многократном обращении к данным в период нахождения их на уровне x_1, дуга b определяет передачу данных с уровня x_1 на уровень x_2 для длительного хранения, а дуга c — передачу данных с уровня x_2 на уровень x_1.

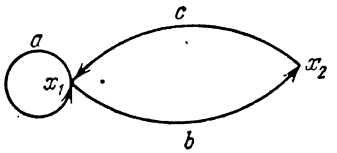
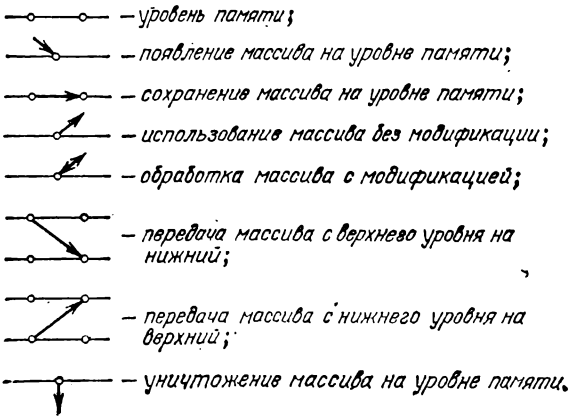


Рис. 1.4. Граф управления данными в двухуровневой системе памяти

Построим модель управления массивом данных в двухуровневой системе памяти на основе рассмотрения возможных состояний массивов на уровнях памяти и переходов массивов из одного состояния в другое.

Введем условные обозначения:



С помощью этих обозначений управление массивом в двухуровневой системе памяти представим в виде диаграммы (рис. 1.5).

Полученный в задаче массив m передается на хранение на уровень x_1 (рис. 1.5, *а*). Здесь массив подвергается многочисленным обработкам, в результате которых он может быть неоднократно модифицирован. По истечении некоторого периода использования массив передается на уровень x_2 (рис. 1.5, *б*). Эта передача

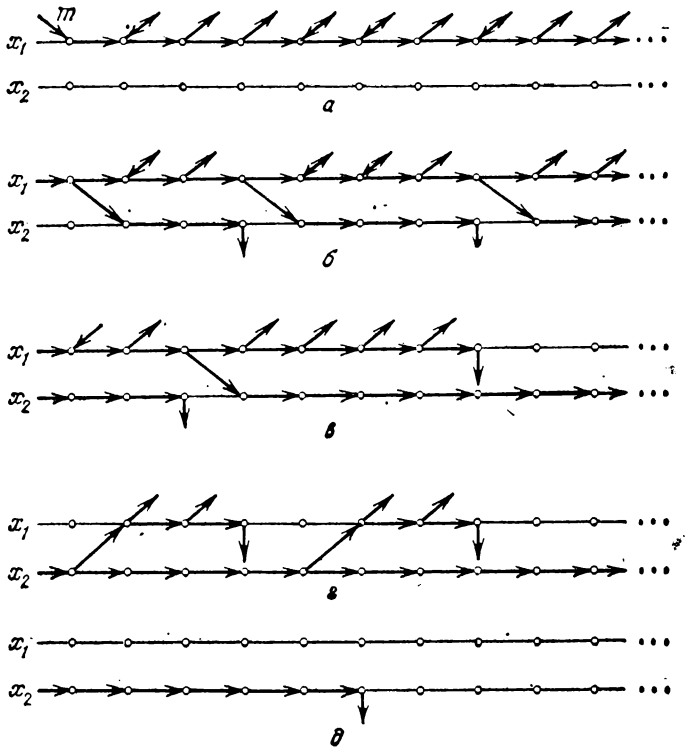


Рис. 1.5. Управление массивом данных в двухуровневой системе памяти

заключается в получении копии массива данных на уровне x_2 с сохранением его на уровне x_1. После завершения передачи массив, сохранившийся на уровне x_1, может быть использован и снова модифицирован. По мере накопления изменений может быть выполнена повторная передача массива с уровня x_1 на уровень x_2, при этом ранее полученные копии на уровне x_2 уничтожаются (рис. 1.5, *б*).

Частота передачи массива с верхнего уровня памяти на нижний уровень должна выбираться для каждого множества отдельно. Если к массивам множества предъявляются высокие требования сохранности (например, в программном множестве), то передачу на уровень x_2 следует выполнять после каждого сеанса обработки, в результате которого произошла модификация массива. Для результатных множеств целесообразнее выполнять передачу массива на уровень x_2 ближе к завершению периода активного использования массива, с тем чтобы передача эта выполнялась для каждого массива один раз.

Когда период активного использования массива завершается, массив уничтожается на уровне x_1 и сохраняется только его копия на уровне x_2 (рис. 1.5, *в*).

Если возникает необходимость использования массива, находящегося на уровне x_2, то он автоматически будет передан на уровень x_1 (рис. 1.5, *г*) и снова может быть неоднократно использован. Копирование с уровня x_2 на уровень x_1 может быть выполнено неограниченное число раз.

На уровне x_2 массив остается до истечения срока хранения, после чего он уничтожается (рис. 1.5, *д*).

Рассмотрим функцию $P(\tau, t) = \sum_i v_i a_i(\tau, t)$, которая позволяет оценить объем памяти на верхнем уровне, необходимый для обеспечения удовлетворительной обработки информационного потока, вызываемого множеством M.

Заметим, что из неравенства $\tau_1 < \tau_2$ следует $P(\tau_1, t) \leqslant P(\tau_2, t)$, так как $a_i(\tau_1, t) \leqslant a_i(\tau_2, t)$ для всех t.

При $\tau = 0$ функция $P(0, t)$ выражает зависимость объема используемых данных от времени. Если объем памяти на верхнем уровне будет равен $P_0 = \max_t P(0, t)$, то тем самым будет обеспечено хранение массивов только на время их непосредственной обработки. Это означает, что после каждого сеанса работы с массивом его придется перемещать на нижний уровень, а при повторном обращении к нему — на верхний уровень. Число перемещений будет равно числу скачков функции $a_i(0, t)$.

С увеличением τ будет увеличиваться и $P(\tau) = \max_t P(\tau, t)$. При этом будет обеспечиваться задержка массивов на уровне x_1 после их очередного использования на время τ и будет уменьшаться число скачков функции $a_i(\tau, t)$.

При достижении объема памяти $P_L = \max_t P(L, t)$, где $L = \max_i \tau_i$, будет обеспечена такая обработка информационного потока, при которой перепись массивов будет выполняться только с верхнего уровня памяти на нижний уровень.

Учитывая неравномерность использования массивов в течение их активного периода, увеличение объема памяти на верхнем уровне до значения P_L может быть экономически не оправдано. Отсюда возникает задача поиска наиболее приемлемого объема $\bar{P}$ $(P_0 < \bar{P} < P_L)$, при котором обеспечивается достаточно высокая эффективность обработки информационного потока.

Заметим, что нельзя объединять все информационные потоки в один поток и выделять для него общий ресурс памяти, как это показано на рис. 1.6.

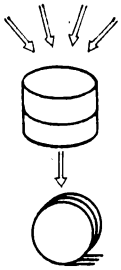


Рис. 1.6. Общий информационный поток

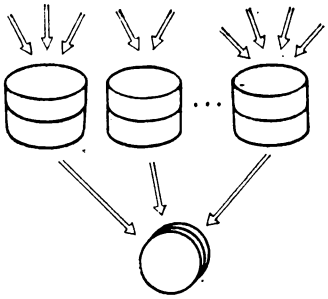


Рис. 1.7. Разделение информационных потоков на верхнем уровне

Такое объединение может привести к необоснованному увеличению памяти верхнего уровня, так как

$$\sum_k v_k a_k (L, t) \gg \sum_j \sum_i v_i a_i (L_j, t),$$

поскольку $L = \max\limits_k \tau_k \geqslant L_j = \max\limits_i \tau_i$, где индекс k пробегает по всем массивам общего информационного потока, а индекс i — по массивам отдельных информационных потоков. Отсюда следует, что система управления данными должна обеспечивать формирование отдельных потоков и предоставлять средства для создания на верхнем уровне отдельных областей внешней памяти.

Следует заметить также, что информационные потоки, имеющие равные периоды активного использования массивов, следует объединять в более крупные потоки, так как это уменьшит число указанных областей и облегчит управление памятью (рис. 1.7). Память верхнего уровня должна быть разделена между областями таким образом, чтобы нагрузка на все области была по возможности одинаковой.

Описанное разделение информационных потоков позволяет повысить эффективность использования памяти верхнего уровня,

но не решает проблем, возникающих при управлении памятью нижнего уровня. Так, для снижения нагрузки на операторов ЭВМ заполнение памяти нижнего уровня нужно осуществлять «по кольцу»: выбирать очередной свободный том (магнитную ленту или магнитный диск), переписывать на него вплотную друг к другу отработавшие массивы, при заполнении текущего тома выбирать очередной свободный том и т. д. Резерв свободных томов будет пополняться при этом по мере уничтожения массивов.

Однако объединение информационных потоков при передаче данных с верхнего на нижний уровень памяти приведет к тому, что к моменту заполнения текущего тома ни один из ранее заполненных полностью не освободится, в то же время будут иметься тома с малым объемом хранимых данных. Причина возникновения такой ситуации заключается в размещении на одном томе массивов с сильно отличающимися сроками хранения.

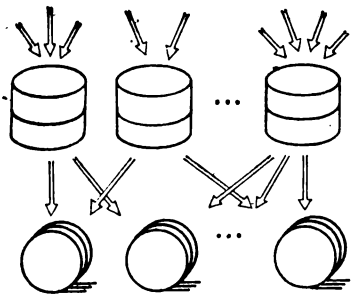


Рис. 1.8. Перегруппировка информационных потоков при передаче массивов данных на второй уровень памяти

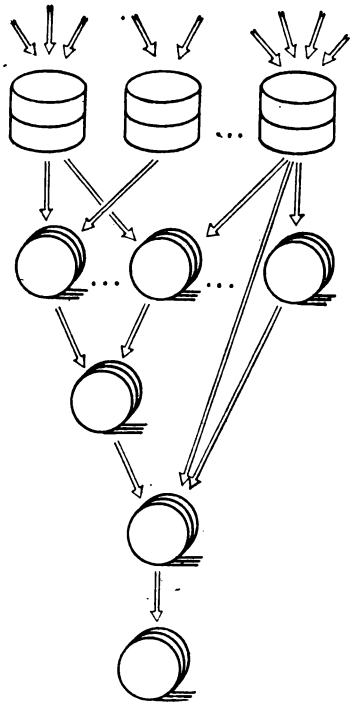


Рис. 1.9. Организация многоуровневой памяти

Дальнейшее развитие механизма управления данными состоит в перегруппировке информационных потоков при передаче массивов с верхнего уровня на нижний и разделении общей памяти нижнего уровня на несколько областей. Это разделение может быть выполнено с разной степенью дифференциации. Можно, например, выделить отдельную область памяти каждому инфор-

мационному потоку, можно выделить область памяти на группу информационных потоков, имеющих одинаковый срок хранения массивов, и т. п. (рис. 1.8). Следовательно, система управления данными должна предоставлять средства для создания областей памяти и на нижнем уровне.

Поскольку характеристики информационных потоков могут изменяться с течением времени, может возникнуть необходимость в перегруппировке потоков как на верхнем, так и на нижнем уровне памяти. При этом управление информационными потоками, очевидно, должно обладать достаточной гибкостью и эффективностью.

Информационные потоки формируются пользователями, и не исключено, что в информационный поток попадет массив, отличающийся по своим свойствам от характеристик данного потока. Выявление таких массивов и отделение их от потока может быть реализовано путем организации многоуровневой памяти (рис. 1.9).

Таким образом, можно рассматривать более общую модель управления данными (рис. 1.10). Вершины графа x_i — уровни памяти, состоящие из одной или нескольких локальных областей.

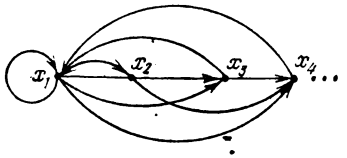


Рис. 1.10. Граф управления данными в многоуровневой памяти

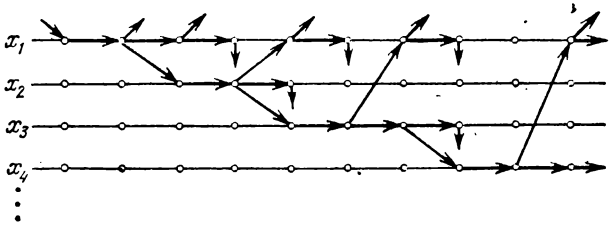


Рис. 1.11. Управление массивом данных в многоуровневой памяти

Множество дуг, направленных из вершины x_1 в вершины x_i ($i \neq 1$), определяют передачу массивов с верхнего уровня на более низкие уровни памяти для хранения, а дуги, направленные из вершин x_i ($i \neq 1$) в вершину x_1,— передачу массивов данных на верхний уровень для обработки. Дуги, направленные из вершин x_i в вершины x_j ($1 < i < j$), определяют передачу мас-

сивов на более низкие уровни памяти. Такая передача выполняется в тех случаях, когда массив какого-либо информационного потока мешает заполнению томов «по кольцу», т. е. при передаче массива с уровня x_i ($i \neq 1$) на более низкий уровень массив на прежнем уровне не сохраняется (рис. 1.11).

1.3. Модели систем управления данными

Проблема выбора наиболее рационального размещения данных в памяти ЭВМ является одной из основных при организации хранения данных. Исследование проблемы привело к разграничению понятий, относящихся к категориям данных и памяти, а затем к их полному разделению на логическую структуру данных, которую должен понимать пользователь системы, и физическую структуру хранения, которая скрыта от пользователя [37].

Очевидно, что способ отображения структур данных в структуры хранения во многом определяет эффективность систем управления данными. В связи с этим в подобных системах разрабатываются средства, обеспечивающие наиболее экономичные структуры хранения при обработке данных различных логических структур. Можно выделить два важных круга вопросов, возникающих при решении названной проблемы. Первый связан с созданием средств, реализующих структуры хранения, наиболее часто используемые в прикладных задачах. Как правило, этот аспект связан с разработкой структур хранения на уровне файла. В этом направлении достигнуты значительные результаты. Системы управления данными имеют богатые по возможностям библиотеки программ методов доступа, позволяющие организовать данные различными способами на разных устройствах внешней памяти. При этом эффективность обработки данных во многом зависит от пользователя, так как именно он осуществляет выбор структуры хранения при решении конкретной задачи.

Второй круг вопросов заключается в решении задачи рационального размещения всей информационной базы вычислительного центра на всех ресурсах внешней памяти. Приведенный в п. 1.1 анализ информационной базы свидетельствует о том, насколько важно при этом учитывать конкретные характеристики информационных потоков.

Для поиска возможных решений проблемы рационального размещения всей информации во внешней памяти ЭВМ проведем анализ систем управления данными с использованием простых моделей. На рис. 1.12 представлена модель системы управления данными S, которая осуществляет передачу данных D между пользователем или его задачей и внешней памятью. Система S обеспе-

чивает передачу данных D и размещение их в неоднородной внешней памяти согласно некоторой функции R.

Выделим в системе S два модуля: модуль передачи данных (М1) и модуль управления размещением данных во внешней памяти (М2) (рис. 1.13). Модуль передачи данных обеспечивает различные способы передачи данных между оперативной памятью

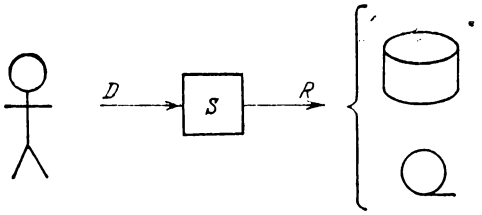


Рис. 1.12. Модель системы управления данными

задачи и внешней памятью ЭВМ, а модуль управления размещением данных определяет адреса внешней памяти, по которым следует располагать данные и с которыми необходимо связать задачу пользователя при последующей обработке этих данных. Следовательно, в модуль М1 поступает информация двух различных видов: собственно данные D, подлежащие хранению, и управляющая информация U от модуля М2. Аналогично можно выделить и два вида информации, поступающей в систему S: данные D и управляющая информация Q, на основании которой вырабатывается управляющее воздействие.

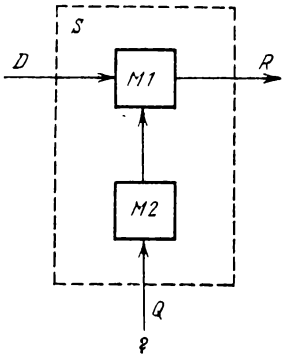


Рис. 1.13. Схема взаимодействия модулей М1 и М2

Что является источником управляющей информации Q? На разных этапах развития систем обработки данных ответ на этот вопрос был разным и соответственно приводил разработчиков к различным по возможностям и по организации системам управления данными.

В ранних системах источником управляющей информации был сам пользователь. При этом системы управления данными были организованы на принципе прямой связи и несли небольшую функциональную нагрузку. В системах с прямой связью (рис. 1.14) управляющая информация Q представляет собой часть всей информации, поступающей от пользователя в систему, и полностью определяет размещение данных D в памяти ЭВМ.

Примером реализации такой модели может служить организация прямого доступа к томам внешней памяти [81, 82]. Модуль М2 в данном случае обеспечивает привязку задачи и выданной из нее заявки к абсолютному адресу внешней памяти, а функция модуля М1 заключается в выполнении физического обмена между оперативной памятью задачи и указанной областью внешней памяти.

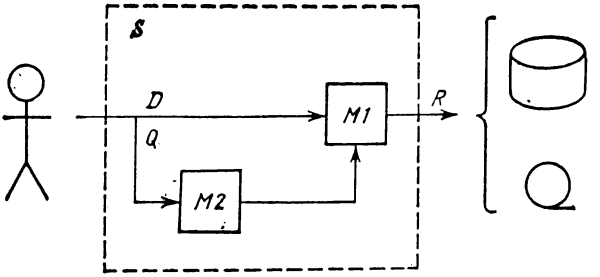


Рис. 1.14. Модель системы управления данными с прямой связью

Такой способ управления большим объемом данных и большими ресурсами внешней памяти становится трудоемким и приводит к возникновению ошибок. Поэтому возникла необходимость создания систем, в которых функция учета данных и размещения их в выделенной области внешней памяти была бы полностью автоматизирована. Такие системы управления данными можно отнести к классу специализированных систем.

Локальные области внешней памяти и данные, размещаемые в них, часто рассматривались как автономные архивы данных. Функция управления данными рассматривалась как внутренняя функция программы, обрабатывающей эти данные. Такие системы были ориентированы на конкретные приложения и, как правило, эффективно решали проблему размещения данных в выделенных им областях внешней памяти. Выбор рационального размещения данных во внешней памяти в этих системах основан на использовании обратной связи (рис. 1.15).

В системе с обратной связью пользователь передает данные и информацию только о их логической структуре, а система в соответствии с некоторым показателем эффективности P обеспечивает наиболее рациональное размещение данных в выделенной области внешней памяти.

Накопленный со временем опыт разработки и эксплуатации таких систем показал, что в пределах «ограниченных решений» проблемы управления данными каждый раз приходится заново решать задачи, общие для различных архивов данных.

Дальнейшее развитие систем управления данными идет по пути создания систем с базовыми возможностями. С функциональной точки зрения это означает, что определяется некоторый базисный набор функций, реализация которого позволяет удовлетворить общие требования достаточно широкого круга пользователей. Что же касается специальных и дополнительных требований, то они должны обеспечиваться с помощью пакетов прикладных программ, ориентированных на конкретного пользователя, или с помощью средств программирования, предоставляемых пользователю.

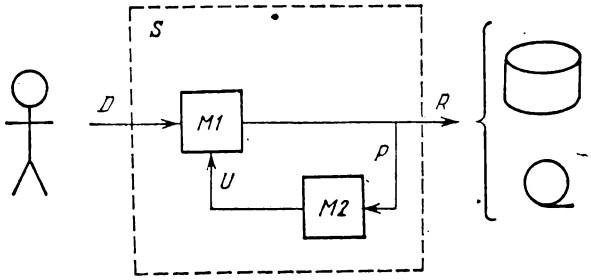


Рис. 1.15. Модель системы управления данными с обратной связью

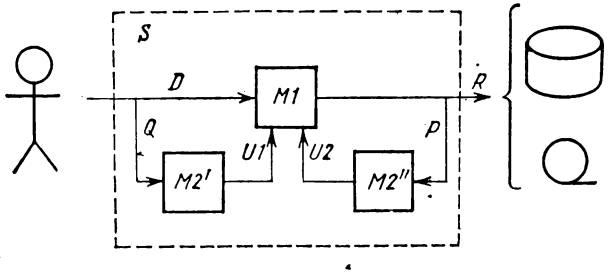


Рис. 1.16. Модель системы управления данными с базовыми возможностями

В системах с базовыми возможностями реализуется модель, в которой функция размещения данных в памяти ЭВМ разумно разделена между пользователем и системой. Традиционной в этом смысле является модель, приведенная на рис. 1.16. Модуль M2′ получает от задачи пользователя сведения о размещении данных в некоторой ограниченной области памяти, а модуль M2″ контролирует размещение данных в пределах указанной области.

Например, в OS/360 [27] пользователь указывает том или группу томов, на которых необходимо разместить новый набор данных для длительного хранения. Пользователи УПД-6 [30, 31, 59, 60] направляют наборы данных на хранение либо в локальный банк данных (организованную совокупность общих томов внешней памяти), либо указывают нужные номера томов. Пользователи этих систем могут не знать точного размещения наборов данных, так как функция размещения данных в указанных областях внешней памяти автоматизирована.

Системы с базовыми возможностями существенно облегчили работу пользователей, но сохранившаяся за каждым пользователем функция распределения данных по физическим областям внешней памяти препятствовала рациональной организации всей информационной базы.

Дальнейшее развитие систем управления данными связано с устранением этого недостатка и сохранением преимуществ систем с базовыми возможностями. Это означает, что пользователи должны работать только с логической структурой данных, а изменения в структуре хранения не должны сопровождаться разработкой новых или изменением существующих программ пользователей. Участие человека в настройке системы не исключается, однако этим человеком не должен быть пользователь. Модель системы, удовлетворяющая этим требованиям, является естественным развитием вышеописанных моделей и находит применение в системах управления базами данных [18, 19, 37, 41—44] (рис. 1.17).

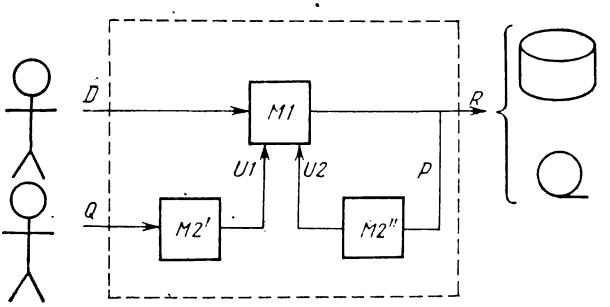


Рис. 1.17

Таким образом, решение проблемы рационального размещения данных в памяти ЭВМ основано на полном отделении пользователя от физической структуры хранения и предоставлении доступа к данным на логическом уровне (по именам) при помощи специально разработанного языка управления данными. Пользователь освобожден от задания сведений, относящихся к размещению данных. Ему предоставляется возможность лишь описы-

вать логическую структуру данных и программировать доступ к данным на логическом уровне. Вопросами организации хранения данных на физическом уровне занимается другой человек — администратор данных.

Следуя [37], термины «логический» и «физический» будем использовать для указания различных аспектов в управлении данными. Термин «логический» будет связываться с представлением и управлением данными с точки зрения пользователя, а термин «физический» будет связываться с реальным хранением данных в памяти ЭВМ и организацией (манипулированием) данных с точки зрения администратора. Одна из главных функций систем управления данными состоит в отображении логической структуры данных в физическую.

Администратору данных должны быть предоставлены средства, с помощью которых он может организовать структуру хранения, наилучшим образом отвечающую приложениям, характерным для данного вычислительного центра. Сервисные средства системы управления данными должны предоставить ему возможность оценивать эффективность выбранной им структуры хранения.

В рассматриваемой модели работа администратора обеспечивается модулем M2′. Модуль M2″ оптимизирует размещение данных, поступающих в систему от пользователя, в рамках той структуры хранения, которая выбрана администратором данных.

При таком подходе организация хранения данных во внешней памяти ЭВМ полностью отделена от пользователя и становится внутренним свойством системы управления данными, а сами данные — ресурсом, управляемым этой системой.

Адаптация к конкретным условиям эксплуатации будет заключаться в предварительной («грубой») настройке системы, выполняемой администратором данных, и более «тонкой» настройке, выполняемой системой автоматически по мере накопления и обработки статистических данных. Возможная архитектура такой системы представлена на рис. 1.18.

Теперь мы можем сформулировать ряд требований, в общем определяющих архитектуру и функционирование систем управления данными, обеспечивающих эффективное проведение вычислительного эксперимента.

1. Для обеспечения эффективной работы пользователя необходим язык взаимодействия с системой, свободный от задания структуры хранения и содержащий средства формирования информационных потоков, т. е. средства манипулирования данными на логическом уровне.

2. Для выбора наиболее рационального размещения всей информационной базы необходимо введение в систему администра-

тора данных, обеспечивающего выделение уровней памяти, разделение памяти на области и управление информационными потоками, т. е. манипулирование данными на физическом уровне.

3. Для обеспечения эффективной работы вычислительной системы внешняя память ЭВМ должна быть разбита на несколько

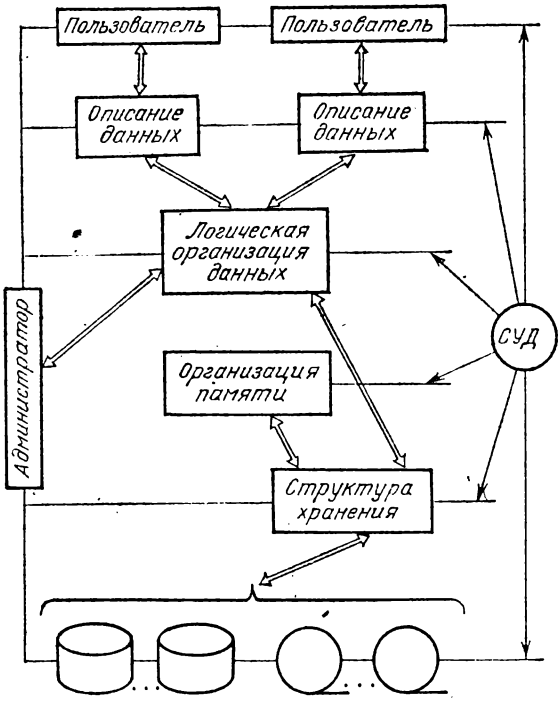


Рис. 1.18. Архитектура системы управления данными

уровней, каждый уровень памяти есть при этом совокупность областей, а информационные потоки должны быть распределены по областям каждого уровня и по возможности сгруппированы с учетом периодов активного использования массивов и сроков их хранения. При этом массивы должны передаваться между уровнями памяти с учетом активности их использования.

Вопросы реализации, функционирования и оценки конкретной системы, в большей степени отвечающей поставленным требованиям, рассматриваются в следующих главах.

ГЛАВА 2

НАДЕЖНОСТЬ СИСТЕМ УПРАВЛЕНИЯ ДАННЫМИ

2.1. Проблемы надежности

При проведении вычислительного эксперимента весьма существенным требованием является достоверность получаемой информации. Уровень выполнения этого требования в основном зависит от степени надежности процессов управления данными, реализованных в той или иной системе.

Обеспечение приемлемой надежности систем управления данными — не только актуальная, но и одна из трудных задач. Это объясняется главным образом сложностью систем, высокой трудоемкостью алгоритмов для работы с различными информационными структурами и сосредоточением в памяти ЭВМ большого количества файлов.

Усложнение систем произошло как в связи с расширением закладываемых в них возможностей для удовлетворения потребностей пользователя и облегчения его работы с файлами, так и с возрастанием функций программного управления внешней памятью для повышения эффективности ее использования. В результате этого расширения увеличилось число элементов системы и связей между ними, появилось также большое количество побочных эффектов, создаваемых этими элементами. Последствия взаимодействия большого числа элементов системы и проявлений побочных эффектов весьма сложно прогнозировать, а отсюда трудно спроектировать систему с точки зрения обеспечения ее высокой надежности. Поэтому сложность называют одной из главных причин ненадежности программного обеспечения в целом [69—72].

Высокая трудоемкость алгоритмов (характеризуемая количеством процессорных операций и операций обменов с внешней памятью), где основные затраты времени дают операции обменов, значительно повышает чувствительность алгоритмов к различным сбоям и отказам.

При массовых расчетах в памяти ЭВМ накапливается и хранится большое количество файлов. В связи с этим возрастает вероятность потери или искажения файлов в результате сбоев или отказов. Такие последствия сводят эффективность систем к нулю независимо от того, кто является истинным «виновником» сбоев или отказов.

Не только для конкретной задачи, но и для вычислительной системы в целом «цена» ненадежности системы управления данными очень высока: ее отказ может привести к отказу ВС.

В данной главе строятся и исследуются модели систем управления данными с точки зрения проблем надежности. На основе анализа этих моделей определяются факторы, влияющие на надежность систем, и места уязвимости систем. Формируются основной подход к решению проблем надежности и основанные на нем методы, средства и реализация. Конечная цель этих рассмотрений состоит в разработке конкретных алгоритмов, приемов и мероприятий для создания высоконадежной системы управления данными, обеспечивающей проведение ответственных расчетов.

Практика показывает, что сбои и отказы систем обусловлены программными ошибками и возмущениями окружающей среды. И те, и другие неизбежны: первые из-за ограниченности существующих средств проверки, тестирования и отладки больших и сложных программных систем, к которым относятся системы управления данными; вторые из-за вероятностной природы сбоев и отказов аппаратуры и искажений входных данных первичными источниками (пользователем, оператором, обслуживающим персоналом). Отсюда следует, что даже создание безошибочной (корректной) системы не может гарантировать ее нормальную работу в реальных условиях. Необходимо обеспечить устойчивое функционирование системы при неисправностях аппаратуры, искажениях входных данных и т. п.

Таким образом, при анализе надежности систем управления данными следует выделить два аспекта: корректность, т. е. способность систем выполнять все необходимые операции и не иметь побочных эффектов, и устойчивость, т. е. возможность правильно выполнять операции, несмотря на неблагоприятное воздействие окружающей среды.

2.2. Основные понятия и определения

С появлением ЭВМ третьего поколения и ВС, в которых в значительной степени (по сравнению с машинами предыдущих поколений) решены вопросы надежности функционирования аппаратуры, основной задачей при создании программного обеспечения стало повышение его надежности. Смещение акцентов было

обусловлено всевозрастающим влиянием программных компонент ВС на ее функционирование и постоянным повышением стоимости их создания и сопровождения. При исследовании проблем надежности программного обеспечения выявились две позиции [70—74]; согласно первой эти проблемы суть те же самые, что и для аппаратуры, из второй следует, что для решения проблем необходимо создать новую дисциплину. В указанных работах отмечается, что обе позиции частично правильны и действительная ситуация лежит где-то посередине. В данной работе развивается подход, состоящий в сочетании методов, применяемых для повышения надежности аппаратуры, и методов, зарекомендовавших себя в практике программирования. Рассмотрим модели функционирования аппаратуры и программной системы.

Первая модель описывает три периода «жизни» аппаратуры [77, 83]: приработку, когда происходит отбраковка конструктивных, технологических и производственных дефектов, нормальную эксплуатацию, характеризующуюся внезапными отказами постоянной интенсивности, и старение, когда появляются отказы возрастающей интенсивности, вызываемые износом. Кривая частоты отказов аппаратуры во времени приведена на рис. 2.1, где Тп — момент окончания приработки; Ти — начало износа; Тд — среднее значение долговечности.

В [83] отмечается, что элементы, прошедшие период приработки, имеют наиболее низкий уровень интенсивности отказов, который сохраняется примерно постоянным. Это характерное свойство аппаратуры проявляется на участке нормальной эксплуатации.

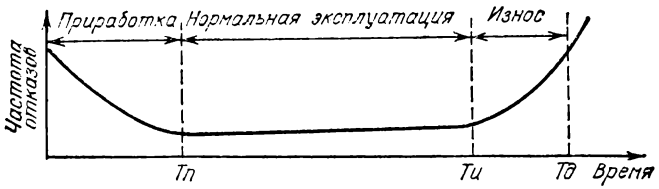


Рис. 2.1. Типичная зависимость частоты отказов аппаратуры во времени

Вторая модель [70, 72—74] характеризует частоту отказов программной системы вследствие ошибок при абсолютной безотказности аппаратуры ЭВМ. Период отладки соответствует периоду приработки, но не по физической сущности (практически отсутствуют ошибки производства, так как встречаются они редко и легко обнаруживаются, например ошибка при копировании системы на ленту). Период нормальной эксплуатации характеризуется отказами вследствие невыявленных при отладке ошибок. При этом частота отказов может возрасти при наличии критиче-

ских условий эксплуатации и разнообразии входной информации. В предположении, что обнаруженные ошибки исправляются, а новые не добавляются, поведение кривой частоты отказов будет таким, как на рис. 2.2 (кривая *а*). Если при исправлении ошибок вносятся «вторичные» ошибки, частота отказов возрастает, что отражено на рис. 2.2 (кривая *б*). В конце концов программная система морально устаревает и подлежит замене. Во второй модели частота обнаружения ошибки фактически зависит от входных данных, ошибки в программе проявляются как систематические, а не случайные события.

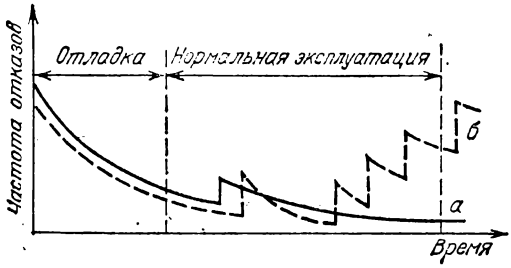


Рис. 2.2. Зависимость частоты отказов программной системы во времени

Следует отметить, что эта модель учитывает также возможность внесения ошибок при развитии программной системы. Так, при эксплуатации системы управления данными могут потребоваться новые функции системы, новая организация данных, метод доступа, могут измениться структуры хранения данных в целях повышения эффективности системы или среда хранения данных в связи с появлением новых запоминающих устройств и т. п. В процессе расширения системы ее отказы так же неизбежны, как и при ее создании. Это подтверждается практикой эксплуатации программного обеспечения [84].

В табл. 2.1 представлены различия проблем надежности аппаратуры и программного обеспечения (по материалам [70—74, 77, 78, 83, 85]). Необходимо добавить к этому, что входные данные для программной системы более разнообразны, чем для аппаратуры; разработчик программ использует более примитивные (по отношению к конечному продукту) «строительные блоки», чем конструктор ЭВМ; технические устройства могут иметь эталонный образец, с которым можно сравнить аналогичные, вновь изготовленные.

Сходства проблем надежности программного обеспечения и аппаратуры состоят в следующем:

— отказ элемента может вызвать отказ системы в целом;

Таблица 2.1

Различия проблем надежности аппаратуры и программного обеспечения

Признак	Аппаратура	Программное обеспечение
Факторы, определяющие надежность системы	1. Надежность элементов (преобладающий фактор) 2. Ошибки проектирования или изготовления	Ошибки проектирования или изготовления
Природа отказов	1. Конструктивный, технологический, производственный дефект элементов 2. Износ элементов	Конфликт исходных данных и характеристик программ [74]
Частота отказов	Зависит от времени	Является функцией входных данных и состояния программ
Последствия отказов	Разрушение элементов	Получение неверных результатов
Долговечность	Ограничена сроком службы	Срок службы не ограничен (но может устареть морально)
Восстанавливаемость	1. Имеются невосстанавливаемые элементы системы 2. Восстанавливаемые элементы и системы подлежат замене или ремонту Восстановление трудно автоматизируется	Невосстанавливаемых элементов нет Восстановление в принципе может осуществляться программными средствами без вмешательства человека
Воспроизведение «аварийной» ситуации	Проблематично	Возможно
Сохраняемость	Зависит от сроков хранения	Не зависит от сроков хранения
Документированные данные об отказах	Приводятся всегда	Практически отсутствуют

— на аппаратуру и программную систему оказывает влияние окружающая среда;

— перед аппаратурой и программным обеспечением ставится общая задача — обеспечение надежной обработки информации;

— необходимы спецификации, описывающие требования для обеспечения надежности;

— необходимо прогнозирование надежности;

— надежность растет по мере совершенствования процесса проектирования, технологии производства и правил эксплуатации.

Надежностные свойства аппаратуры изучаются теорией надежности [86], где устанавливаются закономерности возникновения отказов и определяются методы их прогнозирования; разрабатываются способы повышения надежности изделий при конструировании и последующем изготовлении, а также приемы поддержания достигнутой надежности изделий при их хранении и эксплуатации; создаются методы проверки надежности изделий и способы контроля надежности. Вопрос о необходимости производства программного обеспечения на индустриальной основе уже поставлен на повестку дня [69—72, 87—91], на практике инженерные идеи уже внедряются в область программирования: создание сложных программных систем включает этапы проектирования, изготовления (написания), тестирования и проверки, документирования и внедрения в эксплуатацию.

Термины и понятия теории надежности с учетом отмеченных выше особенностей были приняты за основу надежностной терминологии в программировании. Рассмотрим те из них, которые понадобятся нам в дальнейшем.

В [91] надежность определяется как «свойство объекта выполнять заданные функции с сохранением во времени значений установленных эксплуатационных показателей в заданных пределах, соответствующих заданным режимам и условиям использования, технического обслуживания, ремонтов, хранения и транспортирования». Распространяя это определение на программную систему, необходимо установить функции и параметры, которым она должна удовлетворять, и период времени, в течение которого она должна сохранять эксплуатационные показатели в соответствии с требованиями технического задания, а также величину ущерба каждого отказа для пользователя.

Надежность является комплексным свойством, которое в зависимости от назначения объекта и условий его эксплуатации может включать безотказность, ремонтопригодность (восстанавливаемость), долговечность и сохранность. Для программной системы имеют смысл только первые два свойства. Безотказность характеризует свойство системы сохранять работоспособность в течение некоторого времени или некоторой наработки [91]. Вос-

станавливаемость характеризует способность системы к обнаружению отказов и устранению их последствий. Это свойство зависит от характеристик системы и условий ее эксплуатации. Безотказность и восстанавливаемость качественно характеризуют степень приспособленности системы к выполнению поставленной задачи. С количественной точки зрения необходимы критерии надежности — признаки, по которым оценивается надежность системы. Количественное значение критерия называют *количественной характеристикой* или просто *характеристикой надежности* [77].

Число характеристик зависит от конкретного типа и области применения системы. В основе характеристик лежат понятия о различимых состояниях системы. При исследовании чаще всего ограничиваются двумя возможными состояниями. *Работоспособность* — состояние, при котором система способна выполнять заданные функции с сохранением значений заданных параметров в установленных пределах [91]. *Неработоспособность* — состояние системы, при котором значение хотя бы одного заданного параметра, характеризующего способность выполнять заданные функции, не соответствует установленным требованиям [91].

В процессе функционирования системы возможен ее переход из работоспособного состояния в неработоспособное и обратно. Первый связан с отказами, второй — с восстановлением. *Отказ* — это событие, заключающееся в нарушении работоспособности, т. е. выходе хотя бы одного заданного параметра за допустимые пределы его изменения (допуск). Применительно к отказу рассматривают критерий, причину, признаки (проявления), характер и последствия [91]. Перечень параметров, их допусков и критерии отказа указываются в технической документации на систему. В общей массе отказов чаще всего преобладают сбои, характеризующиеся кратковременным нарушением работоспособности и достаточно быстрым восстановлением. Для аппаратуры понятия «сбой» и «отказ» различаются степенью разрушения компонент, способом и длительностью устранения нарушения работоспособности, для программной системы — длительностью восстановления (см. табл. 2.1).

Программная ошибка — сочетание команд при их использовании, которое при правильных исходных данных дает результаты, не соответствующие эталонным значениям, заданным технической документацией [70]. Исходные, промежуточные или результатные данные, выходящие за пределы допустимых эталонных значений в заданных условиях, называются *ошибочными* (или *искаженными*) [70].

Восстановление — процесс обнаружения и устранения сбоя или отказа системы (а также их последствий) с целью перевода ее в работоспособное состояние.

Показатель надежности — количественная характеристика одного или нескольких свойств, составляющих надежность системы [91]. В основе показателей надежности лежат оценки наработки объекта. По отношению к ЭВМ в качестве наработки рассматривают продолжительность работы [78, 85]. Такая же интерпретация наработки принята для программных систем. Для них показателями надежности являются:

— *среднее время между отказами* — отношение наработки системы к математическому ожиданию количества ее отказов в течение этой наработки [78] (аналогичный показатель вводится для сбоя — среднее время между сбоями);

— *среднее время восстановления* — математическое ожидание времени восстановления работоспособности [78];

— *коэффициент готовности* — отношение времени пребывания системы в работоспособном состоянии к времени эксплуатации, в течение которого происходили отказы и восстановления [78].

Перечисленные показатели надежности оцениваются по результатам работы системы в течение измеряемого периода.

2.3. Уязвимые компоненты и процессы

Системы управления данными характеризуются общностью выполняемых ими функций, составляющих их компонент и протекающих в них процессов. Поэтому возможен единый подход к исследованию их функционирования с точки зрения проблем надежности, в частности, для выявления источников возмущений, уязвимых компонент и алгоритмов.

Рассмотрим некоторую систему управления данными S в предположении, что она не испытывает внешних возмущений. На ее вход поступают данные пользователя D и управляющие воздействия U (рис. 2.3), обеспечивающие течение внутренних процессов в заранее установленных для них режимах. На выходе системы имеются информация E, с помощью которой осуществляется контроль за протекающими процессами, и данные пользователя D' (хранимые и выдаваемые), достоверность которых характеризует степень надежности функционирования системы. Состояние системы полностью определяется внутренними причинами, а управление ею осуществляется изменениями воздействий U.

В реальных условиях большинство процессов протекает при наличии внешних возмущений F окружающей среды (при аппаратных сбоях и отказах, некачественных входных данных и т. п.). Эти возмущения действуют на саму систему, ее входы и выходы и могут привести к ее дезорганизации; нарушить течение проте-

кающих процессов и достижение желаемой выходной информации. Рассмотренная выше схема функционирования при этом изменяется (рис. 2.4).

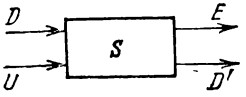


Рис. 2.3. Общая схема системы управления данными при отсутствии внешних возмущений

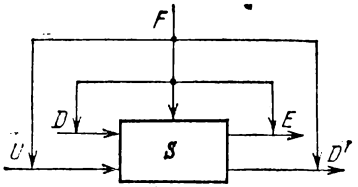


Рис. 2.4. Общая схема системы управления данными при наличии внешних возмущений

Для стабилизации системы необходимо ввести дополнительно компоненту *Z*, подавляющую или нейтрализующую внешние возмущения (рис. 2.5).

Компонента учитывает возможность искажения данных на входах и внутри системы, собирает информацию о состоянии входов, выходов, реагирует на возмущения (а в ряде случаев и на отрицательные внутренние явления) и формирует на основе этого организующие систему сигналы.

Таким образом, функционирование некоторой системы управления данными определяется совокупностью протекающих в ней процессов, наличием влияния окружающей среды и развивающихся внутренних явлений, комплексом управляющих воздействий.

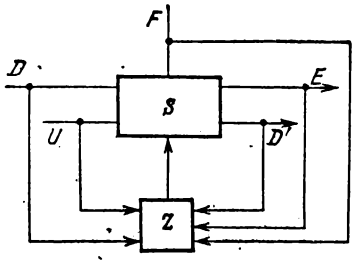


Рис. 2.5. Общая схема системы управления данными при наличии стабилизирующей компоненты

Процесс (наиболее часто употребляемое понятие в области операционных систем) определяется как основная единица работы в системе, например открытие файла, закрытие файла и т. п. В ряде случаев при исследовании функционирования системы удобнее оперировать процессами, чем алгоритмами. В процессах находят отражение использование ресурсов системы и изменения, которые претерпевают некоторые ее компоненты.

Окружение системы составляют те лица, которые взаимодействуют с ней, и компоненты ВС, с которыми прямо или косвенно взаимодействует эта система (рис. 2.6). Фактически на этом рисунке представлены внешние возможные источники сбоев и отказов системы.

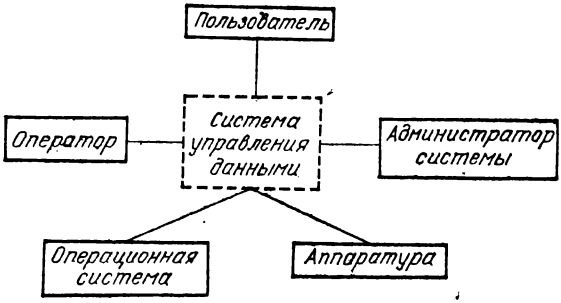


Рис. 2.6. Окружение системы

Пользователь может послать системе некачественные входные данные в операциях доступа к файлам, указать несуществующую операцию, превысить возможности системы (т. е. выйти за рамки правильной работы системы, случайно (непреднамеренно) применить средство ограниченного использования, предоставляемое определенным лицам, ответственным за эксплуатацию системы).

Оператор может проигнорировать или неправильно выполнить указание системы, пропустить ее сообщение о необходимости восстановительных мероприятий, испортить ценную для системы информацию, например, при восстановлении томов внешней памяти.

Администратор системы — лицо, ответственное за ее эксплуатацию, может неправильно сгенерировать систему, допустить ошибки при применении средств ограниченного использования, из чрезмерного доверия к системе выполнять некоторые функции реже, чем требуется, выбрать неправильный способ восстановления системы или ценной для нее информации.

Ошибки ОС, аппаратные сбои и отказы могут привести к искажению информации или нарушению процесса функционирования системы.

2.3.1. Уязвимые компоненты. При работе системы управления данными уязвимыми являются информационные компоненты: служебная (управляющая) информация, описывающая тома внешней памяти, переданные системе, файлы и их местоположение; информационная база (база данных) — совокупность всех файлов. Сами программы (реализующие процессы управления данными и со-

ставляющие в совокупности систему управления) в ходе их исполнения защищены от искажений со стороны «соседей» практически во всех современных ЭВМ и операционных системах [92, 17]. Хранение программ на магнитных носителях при отсутствии внешнего вмешательства характеризуется всегда высокой надежностью.

База данных, как уже отмечалось, уязвима из-за большого количества содержащихся в ней файлов различного назначения и принадлежности, а также сложности различных (последовательных, индексно-последовательных, библиотечных и др.) файлов, буферизации файлов при их обработке и наличия задач, не взаимодействующих с системой и обрабатывающих данные на физическом уровне (т. е. имеющих непосредственный заказ и доступ к томам). Служебная информация отражает состояние базы данных и нередко бывает сосредоточена в одном месте, на так называемом резидентном томе (РТ) системы. В составе служебной информации почти всегда можно выделить основные структурные элементы (объекты): оглавление РТ, таблицу распределения внешней памяти, каталог файлов. Эти объекты чаще других подвержены модификации, перемещаются по РТ, с ними связаны сложные и «длинные» преобразования. Потеря или искажение перечисленных объектов означает потерю или искажение файлов, более того — прекращение функционирования системы.

Таким образом, среди задач обеспечения надежности системы должна быть предусмотрена защита ее информационных компонент от воздействия внешней среды и внутренних возмущающих факторов и регламентированы пути доступа к этим компонентам (рис. 2.7).

2.3.2. Уязвимые процессы. Анализ процессов с целью определения их уязвимости состоит в нахождении точек, где может произойти потеря или искажение информации и отклонение течения этих процессов в результате внешних и внутренних факторов. Как отмечалось ранее, уязвимость информационных компонент системы повышается при их модификации, поэтому круг процессов, связанных с модификацией, значительно сокращается. В наиболее известных системах [29, 30], построенных по идеологии управления данными в OS/360 [27], к таким процессам относятся:

— модификация структуры каталога РТ при вычеркивании набора данных, его переименовании, каталогизации и раскаталогизации, создании и исключении индекса;

— модификация оглавления тома прямого доступа при создании и вычеркивании набора данных;

— запрос пространства для размещения набора данных или его расширения на томе прямого доступа;

— добавление записи в индексно-последовательный (или раздела в библиотечный) набор данных;

— реорганизация индексно-последовательного или библиотечного набора.

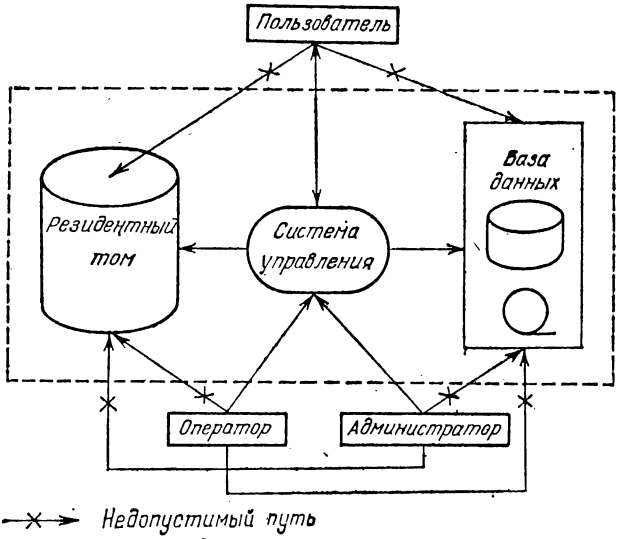


Рис. 2.7. Схема взаимодействия человека с системой

Для иллюстрации нахождения точек уязвимости достаточно рассмотреть процесс создания индекса каталога в ОС ЕС версии 6.1.

Каталог представляет собой специальный набор данных, форматируемый как библиотечный [93]. Создание индекса осуществляется программой IEHPROGM [94]. Пусть имеется индексная структура, представляющая два каталогизированных набора данных К. А. Е. ЕЕ и К. В. С. СС (рис. 2.8). Требуется построить индекс К. В. Х. Y. Индекс Х будет построен автоматически при создании индексного уровня Y, и конечная структура примет вид, показанный на рис. 2.9.

Схему процесса можно представить в виде графа, где вершинами отмечены блоки программы, а дугами — последовательность их выполнения (рис. 2.10). Помеченные цифрами блоки осуществляют: *1* — просмотр индекса тома (высшего уровня каталога); *2* — просмотр индекса К; *3* — просмотр индекса В; *3′* — присоединение новой физической записи в случае заполнения предыдущей в индексе В; *4* — заведение элемента Х в индексе В; *5* — организацию индекса Х; *6* — заведение элемента Y в индексе Х; *7* — организа-

цию индекса Y. На рисунке выделены точки уязвимости (↓), возникающие в результате побочного эффекта связывания физических записей в индексе и самих индексов. Прерывание процесса в данных точках (имеется в виду возникновение аварийной

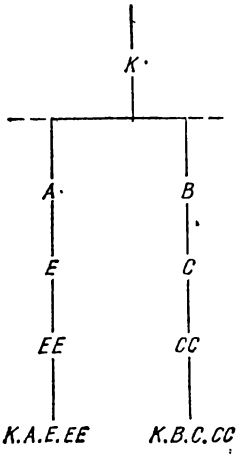


Рис. 2.8. Состояние индексной структуры до создания нового индекса

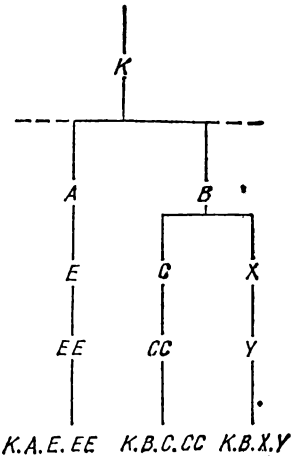


Рис. 2.9. Состояние индексной структуры после создания нового индекса

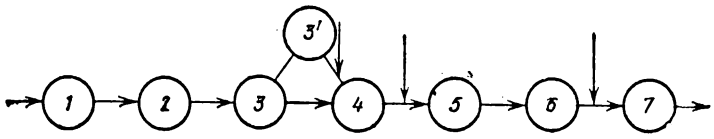


Рис. 2.10. Схема процесса построения индекса

ситуации) может привести к потере записей в индексе, разрыву связей между индексами и в конечном счете к потере наборов данных. Поэтому все связанные с этим процессом преобразования должны выполняться в такой последовательности, чтобы свести к минимуму влияние сбоев и отказов.

2.4. Обеспечение надежности в системах УПД-6 и УПД-ЕС

Вопросам надежности значительное внимание уделено в системах управления данными УПД-6 [30, 60, 95—97] и УПД-ЕС (управление данными в ОС ЕС) [29, 93, 94, 98, 99], наиболее известных в стране систем общего назначения. Первая реализована

как пакет прикладных программ для ОС ДИСПАК [81, 100], вторая является неотъемлемой частью ОС ЕС. Как отмечалось ранее, обе системы по своим концепциям близки к управлению данными в OS/360.

Методы обеспечения надежности в рассматриваемых системах можно разбить на следующие группы:

— предупреждение программных ошибок;

— обеспечение сохранности данных;

— обеспечение сохранности системной (служебной) информации;

— защита от ошибок пользователя;

— обнаружение исключительных ситуаций.

Предупреждение ошибок. Обе системы построены по модульному принципу, используют стандартный интерфейс между модулями; системная информация структуризована, определены допустимые пределы изменения значений полей в информационных структурах; при написании УПД-6 применены макрокод [101] и язык астра [102], а УПД-ЕС — макроассемблер. Все это позволило справиться со сложностью систем и способствовало предотвращению ошибок в готовых программах.

Обеспечение сохранности данных. В системах имеются средства копирования и восстановления наборов данных, осуществляются ведение поколений, контроль структур хранения наборов данных при обработке и копировании.

Обеспечение сохранности служебной информации. Принято различать пассивную и активную формы контроля служебной информации. Пассивный контроль осуществляется в процессе обычной работы системы при выполнении заявок пользователя. Активный контроль предполагает наличие соответствующих средств, которые периодически осуществляют проверку служебной информации с целью своевременного обнаружения ее искажений и принятия мер по их устранению. В УПД-ЕС применяется только пассивный контроль, в УПД-6 — пассивный и активный контроль системной информации локального банка данных [96, 97]. В обеих системах осуществляется копирование служебной информации. Восстановление производится по копии и коррекцией значений на физическом уровне (в УПД-ЕС). В УПД-6 имеются дополнительные средства восстановления системной информации ЛБД [97].

Защита от ошибок пользователя. В системах осуществляется анализ входной информации пользователя, заданной в операторах языка управления заданиями и макрокомандах управления данными, выдаются диагностические сообщения при обнаружении ошибок и коды ответов после выполнения операций. Однако в ряде случаев ошибки пользователя могут привести к непредсказуемым последствиям, например, при попытке открыть блок уп-

равления данными, когда в операнде DCB не задан адрес этого блока, при открытии блока управления данными и использовании набора, когда не было предусмотрено соответствующего оператора DD [99] и др.

Обнаружение исключительных ситуаций. В ОС ЕС предусмотрены средства обнаружения ряда ошибок ввода-вывода и аппаратных сбоев, их устранения или ограничения последствий [103, 104], что позволяет понизить уязвимость процессов управления данными. В этом отношении УПД-6 менее защищена, поскольку ее процессы оформляются в виде прикладных задач, решение которых при возникновении аварийных ситуаций на ВС просто прекращается; меры по ограничению последствий во многом определяются возможностями ОС ДИСПАК, что не всегда бывает достаточно.

В результате проведения рассмотрений систем можно отметить следующее.

При восстановлении системной информации о локальном банке данных в УПД-6 при его профилактическом обслуживании требуется прекращение работы пользователя с банком данных. Это обязывает администратора предпринять ряд организационных мероприятий по удалению задач пользователя из счетных каналов [60], что приводит к значительным потерям времени.

В УПД-6 затруднена защита средств ограниченного использования от применения их лицами, не имеющими на это права. В УПД-ЕС разрешение на применение такого средства дает оператор, которому ОС сообщает, что вызывается защищенная программа. Эта схема разграничения доступа неэффективна: во-первых, затрачивается время на ожидание разрешения от оператора, во-вторых, раскрывается конфиденциальность пароля.

Отсутствие активного контроля служебной информации в УПД-ЕС делает невозможным своевременное обнаружение искажений системных наборов данных. При проявлении этих искажений в процессе работы возможны значительные потери времени, затраченного на решение прикладных задач, завершаемых аварийно, и на устранение некорректностей.

Простое копирование служебной информации не позволяет достаточно полно и точно восстановить ее (в случае искажений) по копии, поскольку последняя устаревает. Регистрация изменений служебной информации значительно улучшает полноту и точность восстановления путем слияния последней копии с изменениями. Однако этот процесс становится сложным и длительным по времени, причем слияние должно осуществляться на некотором промежуточном носителе, чтобы гарантировать его успешное завершение. Большой объем служебной информации только усиливает негативные моменты такого восстановления.

В обеих системах существенна роль администратора в обеспечении контроля служебной информации и ее копирования. При нарушении им регламента этих работ возможны недополучение «свежих» копий и пропуски искажений, что приводит к последствиям, описанным выше.

Таким образом, для этих систем необходимыми предпосылками нормального функционирования являются достаточно надежная работа аппаратуры и ОС и безошибочность работы пользователя, оператора и администратора, что в реальных условиях, как правило, маловероятно.

ГЛАВА 3

ФУНКЦИОНАЛЬНАЯ СТРУКТУРА СИСТЕМЫ УПД ДИАПАК

3.1. Общая характеристика системы

Система УПД ДИАПАК представляет собой совокупность языковых и программных средств, предназначенных для автоматизации управления информационной базой большого объема, возникающей при проведении массовых, взаимосвязанных, сложных физических расчетов на отдельной ЭВМ или на многомашинном вычислительном комплексе, имеющем несколько физических уровней внешней памяти.

Эти средства предоставляют программисту:

— возможность организации длительного хранения и использования данных;

— возможность идентификации, классификации и учета данных;

— единые способы передачи данных на хранение и обращения к ним.

Кроме того, в УПД ДИАПАК решаются задачи:

— ведения единой технологии организации хранения и управления данными;

— предоставления пользователю средств доступа к данным, позволяющих описывать и манипулировать ими только на логическом уровне (по символическим именам);

— обеспечения сохранности данных и защиты областей памяти, занимаемых данными, от взаимных перекрытий;

— обеспечения системы средствами защиты данных от несанкционированного доступа;

— организации совместного использования общих данных;

— эффективного использования имеющегося парка носителей ленточной и дисковой памяти.

Система позволяет организовать хранение индивидуально идентифицируемых массивов информации (файлов) в двухуровневой внешней памяти на магнитных лентах и магнитных дисках,

классифицировать совокупности файлов по различным признакам, выполнять частичное структурирование информации внутри файлов, осуществлять запись и выборку данных, используя различные методы доступа.

УПД ДИАПАК организует высокоэффективную работу вычислительной системы при решении на ней комплекса разнообразных научно-технических задач, обрабатывающих большие объемы данных. Средства управления данными, предоставляемые системой, обеспечивают систематизированные и эффективные способы организации, идентификации, запоминания и выборки этих данных.

Централизованное управление хранением информации позволяет создать единую информационную базу многомашинного вычислительного комплекса, программно объединить магнитные ленты, находящиеся на устройствах всех ЭВМ, определить единую технологию работы с данными.

Двухуровневая организация внешней памяти, автоматическое размещение данных на внешних носителях и перемещение их с одного уровня памяти на другой полностью освобождают программиста от необходимости решения таких проблем, как выделение пространства, выбор типа устройства, определение форматов физического хранения. Ему предоставляется возможность использовать такие категории, как логическая структура данных, объединение массивов данных, осуществление доступа к данным по имени и операции, выполняемые над данными.

Средства, предоставляемые администратору данных, обеспечивают создание локальных областей на каждом из двух уровней внешней памяти и распределение информационных потоков по этим областям. Интерфейс между логическим уровнем управления данными и выбранной администратором организацией внешней памяти обеспечивается системой автоматически.

Общение пользователя с системой осуществляется с помощью языка управления данными [59, 60]. Инициатором диалога может быть как пользователь, так и система. Со стороны пользователя — путем ввода в систему процедур управления данными, выполнением операций записи и выборки данных, выдачей директив с терминала. Со стороны системы — путем подачи сигналов при наступлении некоторых событий, выдачей диагностических сообщений на запросы пользователя, распечаткой директив на операторский терминал для обслуживающего персонала. В диалоговом режиме предусмотрена консультационная помощь в вопросах взаимодействия пользователя с системой.

Система допускает обработку в одной задаче практически неограниченного количества файлов, а значит, и совокупности данных произвольного объема. При этом не требуется установка всех

томов внешней памяти, на которых находятся обрабатываемые данные, перед включением задачи в решение. Закрепление тома за задачей выполняется динамически по ходу решения задачи и только при действительном обращении к нему.

Для управления данными ограниченного использования предоставляются средства защиты, используя которые можно предотвратить посторонний доступ к этим данным.

Управление доступом к данным, обеспечиваемое системой регистрации и учета, позволяет обращаться к данным только по символическим именам без указания местоположения на томах внешней памяти, что уменьшает степень ручного вмешательства и сводит к минимуму как ошибки программиста, так и ошибки оператора ЭВМ.

Используя каталог, можно идентифицировать как отдельный файл, так и совокупность файлов, связанных общим внешним именем, применять к этой совокупности файлов некоторые общие алгоритмы управления, такие как размещение данных в локализованной области внешней памяти, автоматическую «чистку мусора» при исчерпании лимита памяти и т. п.

Независимо от того, где организовано хранение данных (на МЛ или на МД), пользователь реализует обработку данных так, как будто его файл расположен на устройстве с прямым доступом. При этом допускаются четыре типа организации файла: последовательный, индексно-последовательный, прямой и организация разделами.

Справочное обслуживание пользователя обеспечивается комплексом программ, объединенных в подсистему УНИПАК — унифицированный пакет обслуживающих программ [67], с помощью которой можно оперативно получать справки, инициировать генерацию отчетов, просматривать данные, модифицировать их и т. д. При этом результаты могут быть выданы как на терминал, так и на листинг в удобной форме. Здесь же предоставлены пользователю средства копирования файлов, контроля целостности структур хранения на уровне файла, восстановления файла, изменения значений некоторых его атрибутов и т. д.

Для получения характеристик функционирования системы автоматически осуществляется накопление статистической информации. Разнообразие видов обработок накопленной статистики позволяет проводить анализ работы системы с целью выявления «узких мест» в выбранной структуре хранения данных на имеющихся ресурсах внешней памяти, облегчает выбор оптимальных условий функционирования вычислительной системы.

Аппарат оперативного и профилактического контроля за функционированием системы, средства обеспечения сохранности дан-

ных от случайного уничтожения или от последствий сбоев ВС [65] повышают надежность и способствуют успешной эксплуатации системы.

Перечисленные возможности позволяют охарактеризовать УПД ДИАПАК как систему, подходящую для управления данными при решении научно-технических задач, как систему, на базе которой может быть организовано хранение данных с учетом особенностей конкретных условий ВЦ, с учетом возникающих при этом информационных потоков и наличия в конфигурации ЭВМ разнотипного состава внешних запоминающих устройств.

Задачи пользователя, написанные с использованием средств и возможностей УПД ДИАПАК, могут быть решены как в пакетном, так и в диалоговом режимах под управлением ОС ДИАПАК [11] на ЭВМ БЭСМ-6.

3.2. Организация данных

Управление данными состоит в организации логической структуры данных, размещении данных в памяти ЭВМ, передаче данных между основной и внешней памятью. При рассмотрении этих вопросов будем различать единицы данных и единицы памяти. Вводимые ниже понятия соответствуют терминологии, принятой в области обработки данных, учитывают особенности аппаратуры БЭСМ-6, ее внешнего оборудования и физических методов доступа к внешней памяти. Эти понятия представлены в физической и логической классификации.

Физическую классификацию составляют следующие понятия:

том — стандартная единица внешней памяти, используемая для хранения информации,— это бобина МЛ или пакет МД;

блок — единица обмена между основной и внешней памятью,— это зона МЛ или сектор МД.

Размер этих единиц и отношения между ними (по номеру получить блок МЛ или МД) заложены в аппаратуре БЭСМ-6. Размер блока фиксирован. Группу блоков на последовательных адресах тома будем называть *экстентом*.

Логическую классификацию составляют следующие понятия:

запись — совокупность данных, обрабатываемых как одно целое и описывающих некоторый объект;

файл — имеющая имя совокупность записей, объединенных в целое по некоторому организующему правилу;

множество — совокупность файлов некоторого назначения, имеющая общее имя и использующая общий ресурс памяти.

Понятия «запись», «файл», «множество» являются базисными понятиями, отражающими представления пользователя о данных, находящихся на хранении. Связь между ними такова:

— объявляется множество, запрашивается под него ресурс памяти;

— в множестве определяются (создаются) файлы;

— обработка файла заключается в организации обмена записями между файлом и оперативной памятью, занимаемой задачей.

Таким образом, запись представляет собой минимальную единицу данных, с которой имеет дело программа, обрабатывающая файл, и служит логической единицей обмена между оперативной памятью и внешней запоминающей средой. Записи в файлах идентифицируются либо по номеру, либо по ключу. Между записями устанавливается логический порядок следования, который может не совпадать с физическим порядком следования записей в памяти ЭВМ.

Содержание файла может быть любым: текст программы на языке программирования, оттранслированные программы, начальные данные задачи, результаты задачи и т. п. Имя присваивается каждому файлу независимо от его содержания и характера использования.

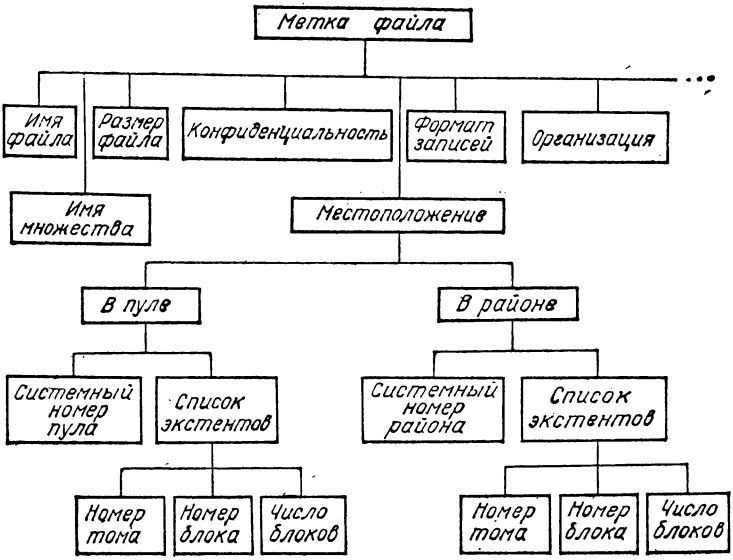


Рис. 3.1. Схема метки файла

Все имена файлов регистрируются в каталоге системы, а атрибуты файла (формат записей, организация, конфиденциальность и т. д.) — в метке файла (рис. 3.1). Средства ведения каталога позволяют регистрировать модификации файлов, классифицируя их как последовательность поколений.

Понятие множества дает возможность классифицировать файлы в соответствии с требованиями пользователя (например, по задачам, по пользователям, по назначению, по характеру использования и т. д.).

Имена файлов, объединенных в одно множество, должны быть уникальными. Два одинаковых имени файла, зарегистрированные под разными именами множеств, идентифицируют разные файлы. Таким образом, обеспечивается однозначный поиск файла, если известны имя множества и имя файла.

Пользователь, объявивший множество, становится его «владельцем». Он назначает атрибуты множества:

— имя;

— ресурс памяти (лимит);

— список пользователей, допущенных к множеству;

— ключ защиты от случайного или преднамеренного уничтожения;

— способ разгрузки множества при исчерпании выделенного ресурса памяти.

Атрибуты множества объединяются в таблицу — метку множества (рис. 3.2), которая, подобно метке файла, регистрируется в системном каталоге под именем множества.

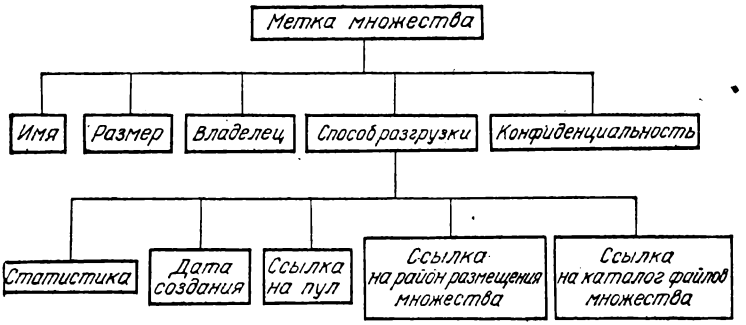


Рис. 3.2. Схема метки множества

По имени множества осуществляется доступ к ресурсам множества и его файлам. Ресурс памяти со временем может быть уменьшен или увеличен. Каждый из допущенных к множеству пользователей может быть наделен полномочиями:

— использования ресурса множества путем создания новых файлов;

— использования других файлов множества, если к ним разрешен доступ;

— уничтожения файлов множества, если ему дано на это право.

Способ разгрузки множества может быть как «ручным», так и автоматическим. При «ручной» разгрузке уничтожение файлов осуществляет сам пользователь. Система обеспечивает защиту от переполнения множества. Если будет осуществляться попытка создать файл сверх максимального ресурса памяти, выделенного для множества, то система снимет задачу пользователя. Автоматическая разгрузка множества служит для своевременного уничтожения устаревших данных (без привлечения пользователя) по одному из алгоритмов:

— уничтожение файлов, срок хранения которых истек;

— уничтожение файлов с минимальным оставшимся сроком хранения;

— уничтожение старых файлов (самых ранних по дате создания).

Системные каталоги имен множеств и файлов представляют собой совокупность иерархически упорядоченных индексов в виде дерева. Количество уровней дерева увеличивается системой автоматически по мере роста числа имен и определяет время поиска, который начинается с вершины дерева. Листьями дерева являются либо метки множеств, либо метки файлов, содержащие сведения о их месте на томах внешней памяти, и прочие атрибуты.

Каталоги расположены на стационарном томе системы и всегда доступны для поиска. Наличие каталогов и оперативный доступ к ним дают возможность каждому программисту переложить функцию учета данных на систему и осуществлять обращение к ним только по имени.

3.2.1. Файлы. Организация файла зависит от двух факторов: физических характеристик памяти, в которой размещается файл, и способа его обработки. При этом последний включает такие моменты, как возможность работы с данными на логическом уровне, желаемая надежность, время ответа, скорость поиска и т. д.

В системе реализованы четыре типа организации файла:

— последовательная организация;

— прямая организация;

— индексно-последовательная организация;

— организация разделами (библиотечная организация).

В последовательно организованном файле логический порядок следования записей совпадает с физическим порядком следования на внешнем носителе (без учета алгоритмов размещения файлов во внешней памяти). Эта организация наиболее эффективна при последовательной обработке данных. Обработка таких файлов может быть организована с равным успехом и на магнитном диске, и на магнитной ленте.

В файлах с прямой организацией обмен между внешней памятью и оперативной памятью задачи осуществляется непосредственно блоками данных. Прямая организация файла позволяет программисту использовать собственную схему адресации записей и их поиска, т. е. собственную организацию файла. Прямая организация позволяет легко адаптировать «старые» программы к сменившейся операционной обстановке, которая возникает при внедрении УПД ДИАПАК.

Записи индексно-последовательного файла располагаются в порядке возрастания значений ключа, которые присваиваются записям в момент передачи в файл. Логический порядок следования записей не совпадает с их порядком на носителе информации и отражается в специальной области памяти, называемой *областью индекса*. Индекс файла позволяет осуществить как прямой (по ключу), так и последовательный (по возрастанию значений ключа) доступ к любой записи файла. Обработка индексно-последовательного файла осуществляется только на устройстве с прямым доступом. Хранение такого файла может быть организовано и на магнитной ленте.

Файл, организованный разделами, состоит из совокупности записей, разделенных на группы (разделы). Различные разделы файла могут быть с последовательной, прямой или индексно-последовательной организацией. Каждый раздел имеет свое уникальное имя в файле, которое вместе с указателем о местоположении хранится в оглавлении файла. Важным преимуществом файла, организованного разделами, является то, что переключение с раздела на раздел влечет гораздо меньшие накладные расходы, чем переключение с файла на файл. Поэтому организация разделами может найти применение в тех задачах, которые рассчитаны на обработку большого количества поименованных массивов данных.

3.2.2. Структура хранения данных. Способ размещения данных (записей, файлов) в физической памяти (блоках, томах) определяет структуру хранения информации на физическом уровне.

Записи и файлы произвольных размеров размещаются системой на блоках и томах фиксированных размеров. При этом большая по объему запись может потребовать несколько блоков памяти, а небольшие записи могут быть сгруппированы в один блок. Аналогично, файл может быть размещен в нескольких томах или на одном томе может быть размещено несколько файлов.

УПД ДИАПАК предоставляет возможность работы с записями данных двух форматов:

— фиксированной длины (формат Ф);

— переменной длины (формат П).

Размер записей формата Ф является постоянным для всех записей файла. Обработка файла с фиксированной длиной записи

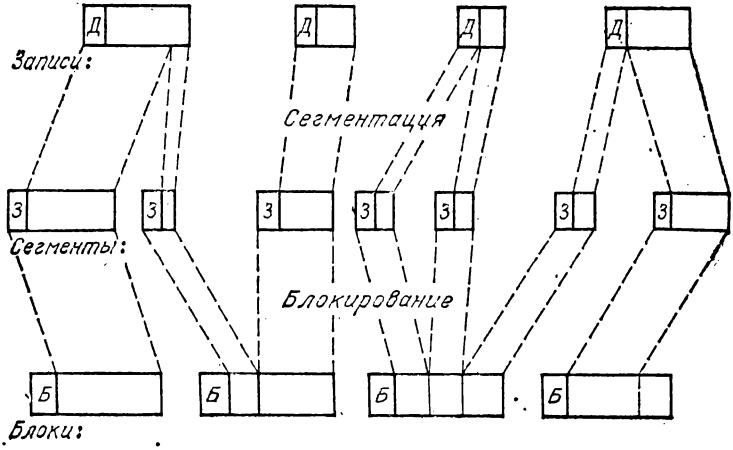


Рис. 3.3. Схема преобразования записей формата П во внутренний формат хранения; Д — длина записи, З — заголовок сегмента, Б — заголовок блока

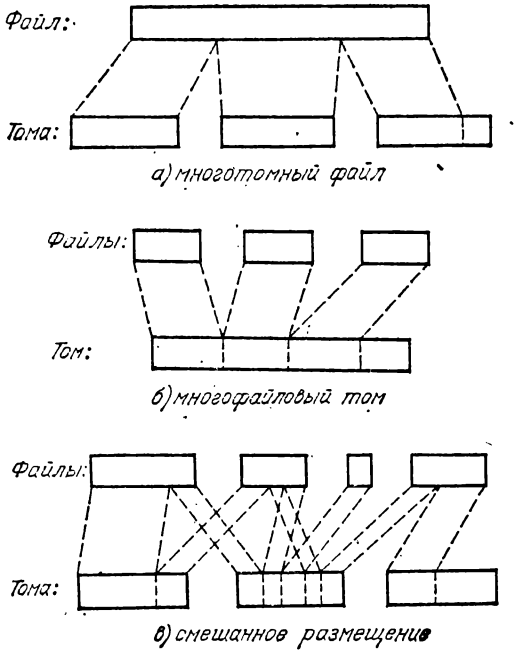


Рис. 3.4. Возможные варианты размещения файлов на томах внешней памяти в УПД ДИАПАК

выполняется в некоторых случаях более эффективно, так как система может вычислить адрес блока внешней памяти, подлежащего чтению, т. е. избежать лишних обменов. Однако использование формата Ф приводит к перерасходу памяти, если не все поля записи содержат данные.

Для более экономного использования памяти может быть выбран формат П. В этом случае размер записи может меняться от записи к записи, тем не менее длина каждой записи должна быть известна и указана при непосредственной передаче ее в файл.

Независимо от формата система проверяет информацию о длине записи и использует ее при дальнейшей обработке файла.

Фиксированный размер блока внешней памяти БЭСМ-6 и произвольная длина записей данных привели к необходимости выбора внутреннего формата физического размещения записей во внешней памяти. Внутренний формат УПД ДИАПАК позволяет обрабатывать и хранить записи любого размера, лишь бы они помещались в оперативной памяти, занимаемой задачей пользователя. Система автоматически переводит любой формат, указанный пользователем, во внутренний формат хранения записей, разбивая их на части (сегменты), объединяя сегменты в блоки в буферной области системы и записывая их на внешний носитель. Блок может содержать несколько записей или сегментов, но не может содержать несколько сегментов одной и той же записи. При добавлении, уничтожении записей и при повторной обработке файла сегментация записей может измениться. Разделение записей на сегменты, объединение их в блоки и обратные манипуляции понятны пользователю, но они неощутимы при программировании.

Выбранный внутренний формат хранения данных сокращает общее число операций ввода-вывода, способствует более полному и эффективному использованию внешней памяти.

Схема преобразования записей формата П во внутренний формат приведена на рис. 3.3.

Задача размещения файлов произвольного размера в томах фиксированных размеров решается в модуле распределения внешней памяти, более полное описание которого будет приведено ниже. Здесь же отметим, что память предоставляется файлу в виде экстентов различного размера. Количество экстентов, выделяемых одному файлу, ограничено. Допускается размещение файла на нескольких томах или нескольких файлов на одном томе (рис. 3.4).

3.3. Организация внешней памяти

3.3.1. Архивное пространство и районы. *Архивное пространство* — это полный ресурс внешней памяти, выделенный для организации хранения данных в системе УПД ДИАПАК. Архивное про-

странство образуется из совокупности томов (бобин магнитных лент и пакетов дисков), прошедших специальную разметку и регистрацию.

Разметка томов осуществляется программно и заключается в определении на носителе участков памяти стандартного размера, достаточного для занесения блоков данных.

Регистрация тома заключается в передаче системе управления данными сведений, характеризующих технические и эксплуатационные свойства тома, и разрешения на его использование. Регистрация осуществляется администратором данных.

Для каждого тома заводится метка тома (рис. 3.5), в которую собираются поступившие при регистрации сведения. Метки томов объединены в единую таблицу регистрации «архивных» томов.

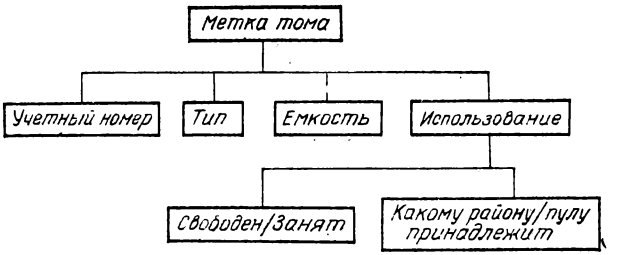


Рис. 3.5. Схема метки тома

Среди всей совокупности томов архивного пространства особое место отведено служебным томам: резидентному, управляющему и фото-томам, которые предназначены для хранения служебной информации (таблиц, меток, каталогов, журналов и т. п.). Остальные тома архивного пространства предназначены для хранения информации пользователя и называются *информационными*.

На основании реально существующего общего объема информационных томов и с учетом заявок на внешнюю память, поступивших от пользователей (отделов, лабораторий, групп или конкретных пользователей), администратор данных составляет бюджет использования архивного пространства и сообщает его системе.

Выделение ресурсов для создаваемых пользователями множеств осуществляется на основании этого бюджета. В ходе эксплуатации можно вносить необходимые изменения, связанные с увеличением лимитов или перераспределением их между пользователями.

На совокупности информационных томов администратор данных создает районы. Район имеет следующие характеристики:

— системный номер;

— тип;

— размер;

— дату создания и др.

Характеристики района объединяются в таблицу — метку района (рис. 3.6). Системный номер идентифицирует область внешней памяти, объединенной в район. Он выбирается автоматически и сообщается администратору данных. В один район могут быть объединены тома только одного типа. Конкретные тома выбираются системой из резерва свободного архивного пространства. Размер района может быть увеличен или уменьшен в ходе эксплуатации. В этом случае система либо наращивает район в допустимых пределах путем подсоединения резервных томов архивного пространства, либо переводит свободные тома района в резерв архивного пространства.

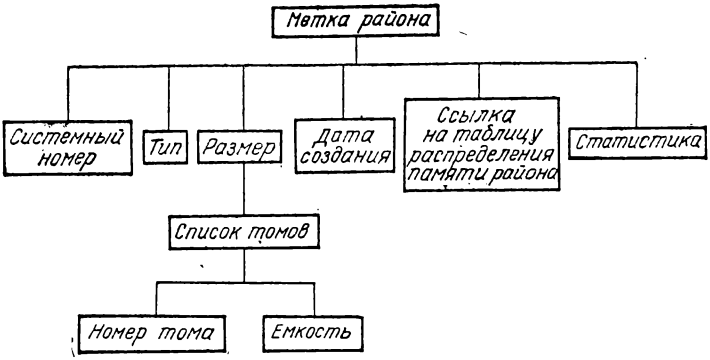


Рис. 3.6. Схема метки района

Районирование архивного пространства можно рассматривать как технический прием для создания двухуровневой организации внешней памяти.

3.3.2. Двухуровневая организация внешней памяти. Двухуровневая организация внешней памяти заключается:

— в создании на верхнем уровне дисковых районов (пулов), являющихся собственным ресурсом памяти системы управления данными;

— в установлении связей между районами и пулами, которые определяют пути передачи файлов;

— в автоматической передаче файлов из пула в районы и обратно.

Связь между районами и пулами устанавливается на уровне имен: в метке района указывается имя пула, с которым данный район связан. Один район может быть связан только с одним пу-

лом, а с одним пулом может быть связано произвольное число районов. Связи между районами, так же как и связи между пулами, запрещены. Районы, связанные с пулами, могут быть как «ленточными», так и «дисковыми».

Легко видеть, что в структуре архивного пространства, полученной путем разделения его на районы, пулы и установления между ними связей, можно выделить два уровня (рис. 3.7): пространство пулов (верхний уровень), пространство районов (нижний уровень).

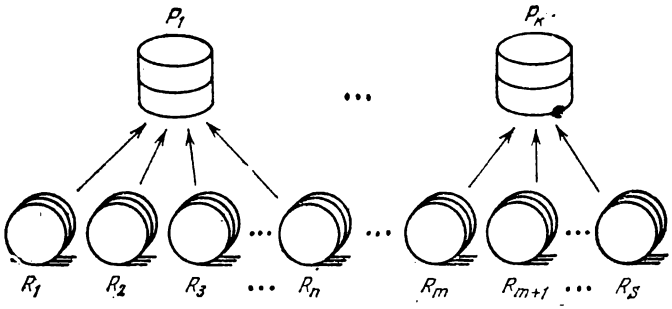


Рис. 3.7. Логическая организация архивного пространства; P_i — пулы, R_j — районы

Связь, установленная между районом и пулом, может быть легко разрушена путем уничтожения имени пула в метке района или заменена на другую связь путем замены имени одного пула на имя другого пула.

Созданием пулов, установлением связей между районами и пулами, разрушением и заменой существующих связей занимается администратор данных. Он же сообщает системе атрибуты пулов, которые запоминаются системой в метках пулов (рис. 3.8).

Связь, установленная между районом и пулом, означает, что файлы, создаваемые в соответствующем множестве, будут размещаться на период активного использования в пуле. При заполнении пула данными будет обеспечена его разгрузка путем передачи файлов в районы. Способ разгрузки пула при заполнении его данными определяется алгоритмом замещения, который подбирается администратором данных для каждого пула персонально и зависит от характеристик информационных потоков, которые проходят через данный пул.

Действия администратора данных по организации внешней памяти не сказываются на программировании и скрыты от пользователя. Это обстоятельство дает администратору данных полную

свободу действий по выбору наиболее рациональной организации внешней памяти и по управлению информационными потоками.

Интерфейс между логическим уровнем управления данными, реализованным в программе пользователя, и организацией внешней памяти, выбранной администратором данных, обеспечивает система управления данными. Возникающие при этом процессы рассматриваются в гл. 4. Здесь же отметим, что перечисленные

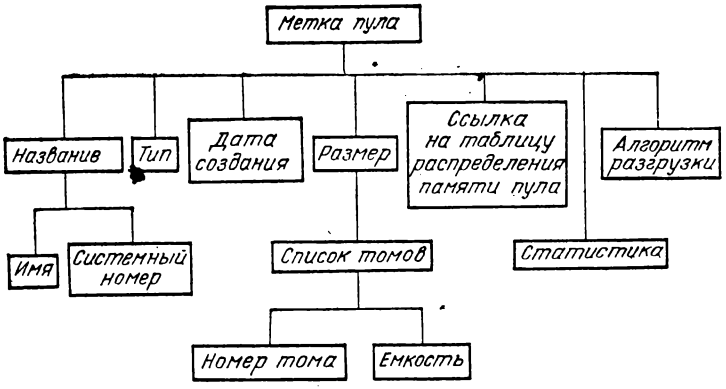


Рис. 3.8. Схема метки пула

возможности двухуровневой организации внешней памяти полностью отвечают требованиям, которые предъявлялись к ней при рассмотрении модели управления данными:

— информационные потоки отделены один от другого;

— память верхнего уровня и нижнего уровня разделена на области;

— на верхнем уровне информационные потоки могут быть сгруппированы;

— при передаче данных с верхнего уровня на нижний уровень потоки разделяются по своим областям внешней памяти;

— администратору данных предоставляются средства, позволяющие выполнять перегруппировку информационных потоков.

3.4. Структура архива

Совокупность томов, районов, пулов, множеств, файлов и связи между ними образуют архив системы управления данными. Определение структуры архива и управление им связаны с организацией большого объема служебной информации, идентифицирую-

щей объекты архива, описывающей их свойства и отношения между ними. Эта информация состоит из меток, каталогов, вспомогательных таблиц и журналов.

Основным источником сведений при создании новых объектов являются директивы и процедуры, задаваемые пользователями и администратором данных. Система автоматически осуществляет построение метки нового объекта и организует ее хранение. За время жизни объекта в его метке накапливается статистическая информация, которая служит источником для выбора оптимальных алгоритмов управления им. Метка объекта уничтожается при поступлении соответствующих директив или процедур. При этом система автоматически разрушает все связи, которые были установлены с объектом.

Каталоги служат для регистрации имен, идентифицирующих каждый из объектов архива. Для ускорения поиска по каталогу он может быть организован по позиционному принципу или в виде дерева. Элемент каталога содержит, как правило, имя регистрируемого объекта и ссылку на метку объекта.

Таблицы используются для размещения сведений, характеризующих объекты, длина таблиц может изменяться. Например, таблица распределения пространства района или пула, которая формируется системой при создании этих объектов, может быть увеличена или уменьшена при изменении размера района или пула. Для такой таблицы заводится ссылка в метках объектов.

Таблицы создаются и для сохранения текущего состояния объектов, находящихся в обработке (например, таблица открытых файлов). Необходимость построения таких таблиц связана с реализацией алгоритмов автоматического восстановления структуры служебной информации после сбоя вычислительной системы.

Журналы используются для автоматической регистрации некоторых событий, возникающих в системе управления данными, с целью анализа этих событий административной службой вычислительного центра. К их числу относятся:

— журнал регистрации попыток несанкционированного доступа к защищенным множествам и файлам;

— журнал регистрации участков внешней памяти, отбракованных из-за дефекта носителя.

Связь между объектами реализуется либо непосредственно путем внесения прямых ссылок из одной метки на другую, либо через каталоги, либо через связующие таблицы. Возможные связи между объектами архива приведены на рис. 3.9. Сложные связи отмечены двойной стрелкой, простые связи — простой стрелкой. В установлении связей принимают участие пользователь, администратор данных и система управления данными, поэтому связи помечены соответствующей буквой П, А или С.

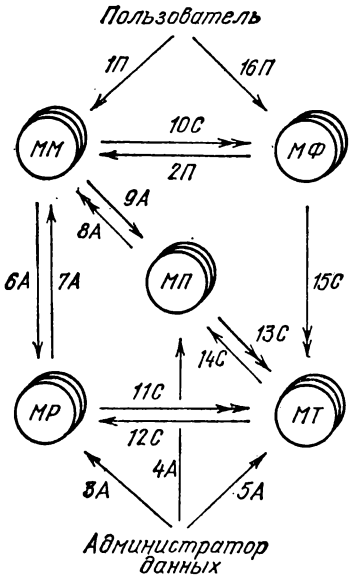


Рис. 3.9. Структура архива УПД ДИАПАК; ММ — метки множеств, МФ — метки файлов, МР — метки районов, МП — метки пулов, МТ — метки томов

Пользователь создает множества (1П) и файлы (16П). Каждый файл он приписывает одному из множеств (2П). Связь 10С используется для генерации отчетов пользователю.

Администратор данных регистрирует тома внешней памяти (5А), создает районы (3А) и пулы (4А). Возникающие при этом связи 11С, 12С, 13С, 14С позволяют формировать отчеты о распределении томов внешней памяти. За каждым множеством администратор данных закрепляет персональный район (6А, 7А), а совокупности множеств, порождающие близкие по характеристикам потоки, он закрепляет за персональным пулом (8А, 9А).

Связь 15С возникает при выделении пространства под файл (для этого используются цепочки связей 2П, 9А, 13С или 2П, 6А, 11С). Она необходима для выдачи запроса на установку тома оператору ЭВМ и закрепления тома за задачей, обрабатывающей файл.

Различные цепочки связей могут быть использованы как для выдачи отчетов или справок, так и для управления архивом.

3.5. Состав системы

УПД ДИАПАК состоит из связанной совокупности модулей. Одни из них реализованы на системном уровне и представляют собой модули операционной системы ДИАПАК, другие являются прикладными программами (рис. 3.10).

По уровню реализации и назначению все модули системы могут быть разделены на три группы (рис. 3.11):

— монитор;
— служебные программы;
— сервисные программы.

Монитор УПД ДИАПАК выполняет базовые функции управления: прием заявок от задач пользователя, управление районами и файлами, распределение внешней памяти, размещение информа-

ции на различных уровнях памяти, ввод-вывод записей и т. д. Необходимость установления тесного взаимодействия монитора со многими модулями операционной системы ДИАПАК [11] (ВВОД, ПЛАНИРОВЩИК, ИНИЦИАТОР, ТЕРМИНАТОР и др.) предопределила его особое место среди других компонент системы. Монитор включен в состав операционной системы ДИАПАК. Взаимодействие монитора с другими компонентами системы управления данными и с задачами пользователей осуществляется через экстракоды операционной системы.

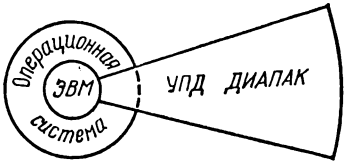


Рис. 3.10. Место УПД ДИАПАК

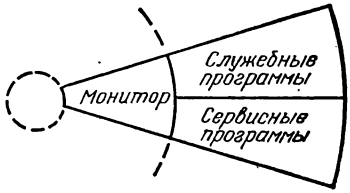


Рис. 3.11. Компоненты УПД ДИАПАК

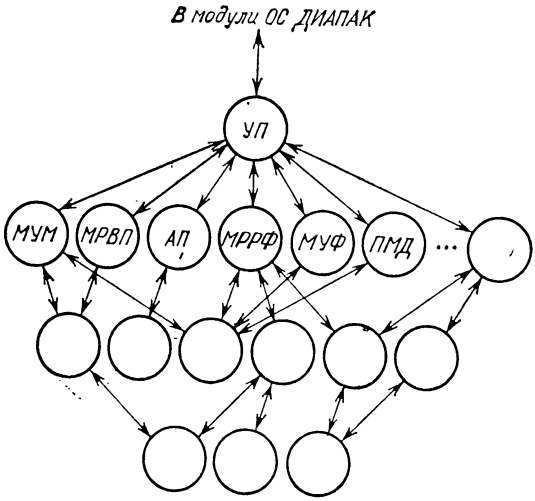


Рис. 3.12. Структура монитора УПД ДИАПАК

Совокупность модулей, составляющих монитор, и связей между ними образует иерархическую структуру (рис. 3.12). Корнем этой структуры является управляющая программа (УП). Она выполняет функции диспетчеризации поступающих в монитор заявок и общего управления их исполнением. Для исполнения заявок она вызывает модули следующего уровня:

— анализатор процедур (АП);

— модуль управления множествами (МУМ);
— модуль управления файлами (МУФ);
— программы методов доступа (ПМД);
— модуль распределения внешней памяти (МРВП);
— модуль рационального размещения файлов по уровням памяти (МРРФ) и т. д.

Эти модули могут вызывать другие модули монитора, относящиеся к более низкому уровню иерархии, и т. д. Загрузка модулей в оперативную память осуществляется по мере надобности и может быть выполнена как по заявке управляющей программы, так и по заявке других модулей монитора.

В связи с функционированием системы на многомашинном вычислительном комплексе и необходимостью выполнения в ряде случаев трудоемких работ, требующих больших ресурсов ВС, некоторые модули реализованы на прикладном уровне. Они названы *служебными программами системы.* В этом случае заявка монитора формируется в виде задачи с соответствующими начальными данными, которая попадает на буфер ввода, общий для всех ЭВМ. Она получает наивысший приоритет и может быть выбрана на обработку любой ЭВМ комплекса.

Служебная задача помечается номером ЭВМ (адресом), на которой она была сформирована, а та ЭВМ, на которой эта задача будет решена, посылает по указанному адресу сообщение о выполнении заявки монитора. Для защиты от сбоев монитор периодически проверяет состояние сформированной им задачи, а в случае обнаружения сбоя повторяет формирование задачи.

Служебные программы выполняют перепись файлов с одного уровня памяти на другой, автоматическую разгрузку множеств и пулов, профилактический контроль функционирования системы, проверку целостности служебной информации, сбор статистики, регулярную обработку и выдачу статистики и т. д.

Совокупность модулей системы, предназначенных для выдачи справок, генерации отчетов, визуального просмотра данных и модификации их с терминала, а также средства ведения архива, предоставляемые администратору данных, образуют *сервисный уровень системы.* Все эти программные средства унифицированы и объединены в единую подсистему, получившую название УНИПАК.

Подсистема УНИПАК, объединяющая все сервисные программы, вызывается пользователем или администратором с терминала. Сервисные средства предоставляются согласно поступившему с терминала паролю. Различные полномочия лиц, пользующихся сервисом системы, находят отражение как в ограничении или расширении возможностей сервиса, так и в содержании выдаваемой информации.

Запрос с терминала инициирует работу соответствующей сервисной программы. Все сервисные программы можно разделить на *диалоговые*, *непосредственно исполняемые* и *пакетные*. Диалоговые программы характеризуются тем, что сами вступают в диалог с пользователем и выполняют его заявки. Непосредственно исполняемые программы получают начальные данные, поступившие вместе с запросом, выполняют требуемые от них действия, полученный результат выдают на терминал или/и на листинг и завершаются. Пакетные программы тоже получают начальные данные, после чего теряют связь с терминалом, приобретают статус самостоятельных задач и поступают на буфер ввода операционной системы. Оттуда они выбираются в решение и выполняют свои функции в пакетном режиме.

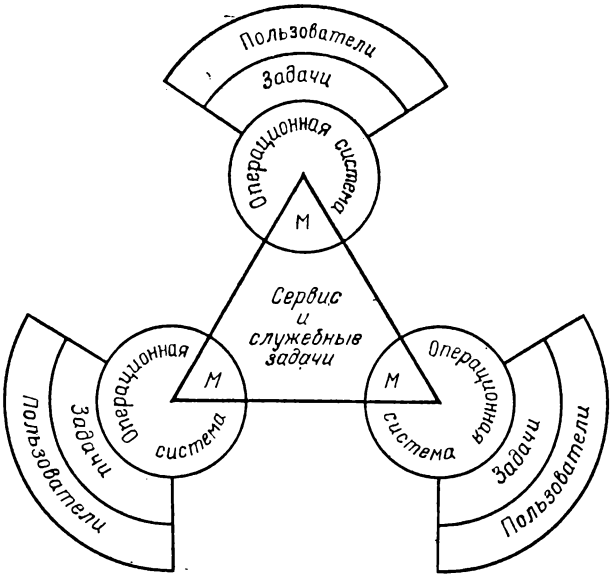


Рис. 3.13. УПД ДИАПАК на многомашинном комплексе; М — монитор, число машин — три

Функционирование УПД ДИАПАК на многомашинном вычислительном комплексе приводит к тому, что монитор системы присутствует на каждой ЭВМ (рис. 3.13), а библиотека служебных и сервисных программ является общей для всех ЭВМ. Служебные программы принимают заявки мониторов и выполняют их на любой ЭВМ, имеющей свободные ресурсы. Подсистема УНИПАК может быть вызвана на любой ЭВМ комплекса.

ГЛАВА 4

ФУНКЦИОНИРОВАНИЕ СИСТЕМЫ УПД ДИАПАК

4.1. Язык взаимодействия

Пользователь организует работу с данными, находящимися на длительном хранении в памяти ЭВМ, используя понятия «множество», «файл», «запись». Он разрабатывает соответствующие алгоритмы и программирует их, используя язык системы управления данными. При работе прикладных программ сведения о данных поступают в систему в виде описаний, совокупность которых дает общее представление о логической организации информационной базы ВЦ.

Язык общения администратора данных с системой позволяет ему получать необходимые сведения о логической организации данных, о структуре информационных потоков, о их характеристиках и т. д. На основании анализа полученных сведений администратор данных вносит необходимые коррекции в организацию внешней памяти.

Конкретная структура хранения выбирается системой управления данными на основании логической организации данных и выбранной администратором организации внешней памяти.

Обмен данными между задачей и внешней памятью выполняется по заявке, поступившей от задачи в систему. При этом используются средства физического обмена с внешней памятью, реализованные в ОС ДИАПАК [12]. Передача данных осуществляется через системный буфер, на котором выполняются операции сегментации, блокирования и обратные им операции. Общая схема функционирования УПД ДИАПАК представлена на рис. 4.1.

Язык системы управления данными включает процедуры управления множествами и файлами, операции обработки записей и директивы. Он обеспечивает связь пользователя и администратора с системой управления данными, отражает все возможности УПД ДИАПАК, обладает достаточной гибкостью и выразительностью.

Процедуры управления множествами и файлами предназначены для взаимодействия пользователя с системой управления данными [66]. Они позволяют:

— определить множество, т. е. присвоить ему символическое имя, объявить список пользователей и их полномочия, установить прочие эксплуатационные характеристики;

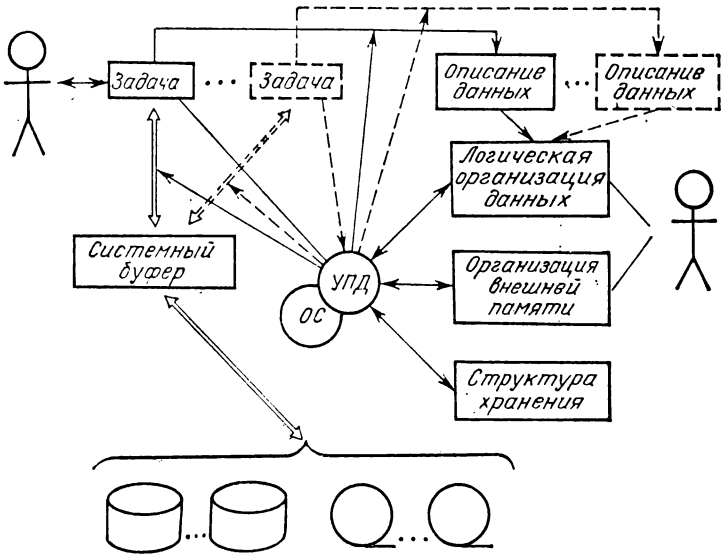


Рис. 4.1. Общая схема функционирования УПД ДИАПАК

— открыть множество, т. е. предоставить доступ задаче для управления файлами множества и/или для создания новых файлов;

— закрыть множество, т. е. прекратить доступ к множеству из задачи;

— уничтожить множество, т. е. уничтожить информацию о названном множестве в соответствующих служебных таблицах, разорвать связи с другими объектами архива и т. д.;

— определить файл, т. е. присвоить файлу имя, указать множество, к которому он относится, максимальный размер, организацию, формат записей, конфиденциальность и другие характеристики;

— открыть файл, т. е. сделать ранее определенный файл доступным для обработки;

— закрыть файл, т. е. прекратить доступ к файлу;

— уничтожить файл, т. е. уничтожить информацию о файле в соответствующих служебных таблицах, освободить занимаемое пространство и т. д.

Процедуры могут быть заданы двумя способами:

— статически (при помощи раздела паспорта задачи [82]);

— динамически (при помощи экстракода).

В каждом конкретном случае способ задания выбирается пользователем и может зависеть от полноты информации об объекте, с которым работает задача, от соглашений, устанавливаемых между задачами в процессе обмена данными, от организации программы или комплекса программ.

Операции обработки записей в системе управления данными предназначены для передачи записей между внешней и оперативной памятью, их модификации или уничтожения [66]. Операции, выполняемые над записями файла и над записями раздела библиотечного файла, совпадают, поэтому далее речь пойдет только об обработке записей файла. Операции обработки записей могут быть разделены на четыре группы:

— создание новых и модификация существующих записей файла;

— выборка записей из файла в оперативную память задачи;

— уничтожение записей;

— операции с блоками данных.

Запись как минимальный адресуемый элемент данных характеризуется *длиной* и *именем*. Именем записи служит *порядковый номер* — в файле с последовательной организацией и *ключ* — в файле с индексно-последовательной организацией.

Порядковый номер записи при внесении ее в файл присваивает система и сообщает его задаче. Ключ записи присваивает пользователь. Размещая записи в файле, система упорядочивает их по возрастанию имен, т. е. создает логический порядок следования записей в файле, который может не совпадать с физическим порядком следования на носителе информации.

Модификация записи осуществляется либо заменой содержимого существующей записи с сохранением или изменением ее длины, либо добавлением к существующей записи некоторого количества данных. Изменение длины записи допускается только для формата П.

Выборка записей может быть выполнена с использованием как их уникальных имен, так и отношения логического следования, установленного при формировании файла. В последнем случае используется указатель на текущую запись. Первоначальная установка указателя происходит при выполнении операции открытия файла и управляется соответствующим операндом процедуры. После выполнения очередной операции обработки файла указатель устанавливается на запись, которая следует (в логическом порядке) за последней обработанной записью. Если обращение к записи выполнено по имени и искомой записи нет, то указатель устанавливается на ближайшую запись с большим именем.

Записи файла могут быть уничтожены как выборочно (по именам), так и группами (начиная с некоторой в логическом по-

рядке следования). Для уничтожения всего содержимого файла без выполнения операций переопределения достаточно выполнить операцию обновления.

Операции с блоками данных предназначены в основном для обработки файлов с прямой организацией. Блоки имеют относительные номера в файле, начиная с нуля. Поблочная обработка допускается и для файлов с любой другой организацией, но в этом случае программист должен знать структуру хранения записей.

Завершение любой операции обработки данных сопровождается выдачей ответа от системы, который может быть положительным, когда операция выполнена полностью, и отрицательным, когда операция задана некорректно или не может быть выполнена в полном объеме.

Диалоговая форма общения пользователя с ЭВМ нашла отражение в языке взаимодействия в виде *директив* [67, 68]. Мнемоника директив, широкое использование правил умолчания, выдача подсказок и подробных диагностических сообщений об ошибках упрощают задание директив и ускоряют диалог.

Язык директив включает:

— приказы инициализации диалоговой формы общения с системой и окончания диалога;

— указания о режиме ведения диалога;

— средства обработки данных с терминала;

— средства ведения архива данных.

Приказы инициализации диалога осуществляют вызов сервисной подсистемы УНИПАК и установление связи между подсистемой и терминалом. Указания о режиме ведения диалога определяют круг полномочий пользователя, находящегося за терминалом, управляют форматом и объемом выдаваемых на листинг документов и сообщений на терминал. Обработка данных с терминала предусматривает исполнение процедур управления множествами и файлами, позволяет визуально просмотреть данные в требуемом формате, модифицировать их, а при необходимости и ввести новые данные. Средства ведения архива данных позволяют запросить справку о состоянии отдельного объекта архива или их совокупности, проконтролировать работу системы, генерировать отчет о содержимом всего архива или его части по некоторым поисковым признакам, внести коррекцию в служебную информацию и т. д.

4.2. Управление данными

В УПД ДИАПАК выделим три уровня управления данными:

— управление информационными потоками;

— управление файлами;

— управление записями.

4.2.1. Управление информационными потоками. Управление информационными потоками является одной из новых функций, реализуемых в системах управления данными.

В управлении информационными потоками принимают участие пользователь (неявно), администратор данных и система.

Пользователь работает с данными на логическом уровне. Он определяет множества, получает к ним доступ на время решения задачи, создает файлы и распределяет их по множествам. Тем самым его участие в управлении информационными потоками заключается в их формировании.

Администратор данных изучает свойства информационных потоков и соответствующим образом организует для них внешнюю память.

Прежде всего, администратор данных создает районы и закрепляет их за множествами. Тем самым он обеспечивает множества ресурсами внешней памяти, в которой будет организовано хранение файлов. С появлением в множестве реального информационного потока администратор данных приступает к изучению его характеристик: анализирует суточную статистику, выдаваемую системой автоматически, запрашивает по директиве с терминала более подробные характеристики исследуемого потока. При необходимости он создает пул для данного потока и устанавливает связь между районом и пулом или соединяет район с существующим пулом, в котором собраны информационные потоки с подобными свойствами. В последнем случае может потребоваться увеличение размера пула.

Со временем свойства информационных потоков могут измениться и прежняя организация внешней памяти может снизить эффективность функционирования вычислительной системы. В таком случае администратор данных изменяет размеры существующих пулов и/или создает новые пулы и выполняет перегруппировку информационных потоков, переключая районы с одного пула на другой. Если необходимость в двухуровневой организации внешней памяти для некоторых информационных потоков отпала, то администратор разрушает связи между соответствующими районами и пулами.

Участие УПД ДИАПАК в управлении информационными потоками заключается в обеспечении автоматического размещения и перемещения файлов в организованной администратором внешней памяти. При этом можно выделить различные функции, выполняемые системой.

Если связь установлена только между множеством и районом, то система управления данными обеспечивает обработку файлов множества непосредственно на томах района. Тома района, как правило, съемные, поэтому система выдает запросы оператору

ЭВМ на установку томов и закрепляет их за задачей пользователя либо в момент включения ее в решение, либо в ходе решения задачи. Если район ленточный, то обеспечивается монопольное закрепление томов за задачей.

При подключении непустого района к пулу система управления данными выполняет постепенную буферизацию активных файлов множества в «пуловском» пространстве. С этой целью она прерывает задачу пользователя в момент обращения к файлу, находящемуся в районе, выдает оператору ЭВМ запрос на установку необходимых томов района, переписывает файл из района в пул, после чего открывает прерванную задачу и обеспечивает обработку файла в пуле. Одновременно с этим память для размещения новых файлов предоставляется сначала в пуле. Постепенно процесс буферизации завершается и информационный поток будет направлен из пула в район, т. е. файлы сначала будут попадать в пул, затем переписываться в район и далее уничтожаться.

Переключение района с одного пула на другой означает, что система управления данными обеспечивает обработку ранее созданных файлов в «старом» пуле, а вновь создаваемые файлы будут размещаться в «новом» пуле. Для того чтобы сократить время нахождения информационного потока в таком «двойственном» состоянии, процесс переключения ускоряется за счет уменьшения времени пребывания файлов в «старом» пуле. При этом могут появиться дополнительные перемещения файлов из района в «новый» пул. Скорость переключения выбирается такой, чтобы снижение эффективности обработки данных было незначительным.

Если администратор данных разрушил связь района с пулом, то система управления данными обеспечивает изгнание из пула всех файлов соответствующего информационного потока. При этом будут инициированы процессы переписи файлов из пула в район и уничтожения их в пуле. В пуле могут задержаться только те файлы, которые в данный момент находятся в обработке. Но и они будут «изгнаны» из пула по ходу освобождения.

4.2.2. Управление файлами. Управление файлом начинается с его определения, т. е. задания атрибутов файла и указания принадлежности его к одному из множеств, предварительно открытых задачей. Система регистрирует имя файла в каталоге указанного множества, формирует метку файла, организует ее хранение. После выполнения процедуры определения файла все обращения к нему осуществляются только по имени.

Выполнение процедуры открытия файла заключается в обеспечении «привязки» задачи к области памяти, на которой указанный файл расположен и на которой будет выполняться его обработка.

Действия системы управления данными при этом состоят в следующем:

— по имени файла в указанном множестве отыскивается метка файла;

— из метки файла извлекается управляющая информация, необходимая для выполнения операций обработки записей, и формируется блок управления файлом;

— блок управления файлом размещается в оперативной памяти системы и закрепляется за задачей пользователя;

— обеспечивается перемещение файла на тот уровень памяти, на котором предусмотрена его обработка;

— сведения о пространстве, выделенном под файл, записываются в блок управления файлом.

Выполнение процедуры открытия файла совмещено с решением задачи пользователя. При этом функция синхронизации может быть возложена программистом либо на систему, либо на задачу. В последнем случае задача узнает о завершении выполнения процедуры либо путем опроса, либо через аппарат событий. Если синхронизация выполняется системой, то задача может быть «закрыта» на достаточно продолжительное время. Основной причиной задержки при этом является перемещение файла из района в пул.

Доступ к файлу может быть предоставлен задаче в монопольном или совместном режиме использования. Если затребован монопольный режим использования, то система обеспечивает единоличное владение файлом в течение всего периода обработки его данной задачей. В совместном режиме использования система допускает к файлу неограниченное число задач с различных ЭВМ комплекса. В этом случае в задачах должна быть предусмотрена блокировка от взаимного влияния при внесении изменений в файл. Этой цели служат средства кратковременного «захвата», позволяющие синхронизировать выполнение задач при прохождении участков программ, модифицирующих файл.

Операция закрытия освобождает файл, закрепленный за задачей. При этом могут быть реализованы два вида освобождения: файл закрывается и уничтожается, файл закрывается и сохраняется.

Файлы, находящиеся на хранении, могут быть перемещены в пределах архивного пространства средствами УПД ДИАПАК без участия пользователя или администратора данных.

Длительное хранение файлов осуществляется согласно установленному пользователем регламенту. Файлы уничтожаются либо пользователем, либо системой. В последнем случае пользователь должен указать алгоритм автоматического уничтожения файлов в множестве.

4.2.3. Управление записями. В УПД ДИАПАК, как и во многих других системах управления данными, имеется набор программ ввода-вывода записей — программ методов доступа. Эти программы осуществляют передачу записей между внешней памятью и оперативной памятью задачи пользователя. Они автоматически выполняют такие функции, как буферизация данных в системной области памяти, сегментация записей, объединение сегментов записей в блоки и выделение записей из блоков. Программы методов доступа используют сведения из блока управления файлом, созданного при открытии файла.

Поиск необходимой записи в файле может быть выполнен двумя способами: просмотром всего файла до тех пор, пока не встретится требуемая запись, или по имени записи. Первый из этих способов доступа является *последовательным*, второй — *прямым*. Выбранный пользователем способ доступа и внутренняя организация обрабатываемого файла определяют метод выполнения ввода-вывода данных.

Для последовательной обработки файла используется логический порядок следования записей в файле. Для обращения к конкретной записи файла или к группе записей используются относительный номер блока в прямой организации, ключ записи в индексно-последовательной организации, номер записи в последовательной организации, имя раздела в организации разделами.

Для поиска записи в файле используются следующие методы:

— последовательное сканирование,

— блочный поиск,

— поиск с привлечением таблиц

и их комбинации. Выбор метода поиска осуществляется автоматически в каждом конкретном случае и определяется организацией файла, возможностями внешнего запоминающего устройства, способом доступа, выбранным для обработки файла.

Метод последовательного сканирования файла является наиболее простым методом и дает наилучший результат для тех приложений, где каждая запись должна быть прочитана и обработана задачей. В прямом способе доступа сканирование файла является процессом, требующим много времени. Тем не менее этот метод нашел применение для точной локализации записи в некоторой небольшой области, если эта область найдена другим методом.

Для нахождения записи в последовательном файле по ее номеру используется блочный поиск в сочетании с последовательным сканированием. При блочном поиске осуществляется переход через несколько записей или через несколько блоков внешней памяти, поэтому его часто называют поиском с переходами [111]. Поиск выполняется до тех пор, пока не будет найдена достаточно малая окрестность искомой записи, после чего либо уменьшается длина

перехода, либо предпринимается последовательный перебор записей. Последовательное сканирование локализованного участка памяти идет до тех пор, пока не будет найдена искомая запись или не будет зафиксировано ее отсутствие.

Рассмотрим алгоритм блочного поиска с переменной длиной перехода, реализованный в УПД ДИАПАК. Пусть последовательный файл содержит N записей (в общем случае переменной длины) и занимает L блоков внешней памяти. Структура хранения обеспечивает в начале каждого блока номер первой (i) и последней (j) записей в блоке. Пусть головка чтения/записи устройства находится в начале M-го блока файла. Тогда поиск искомой записи X осуществляется по алгоритму, блок-схема которого приведена на рис. 4.2.

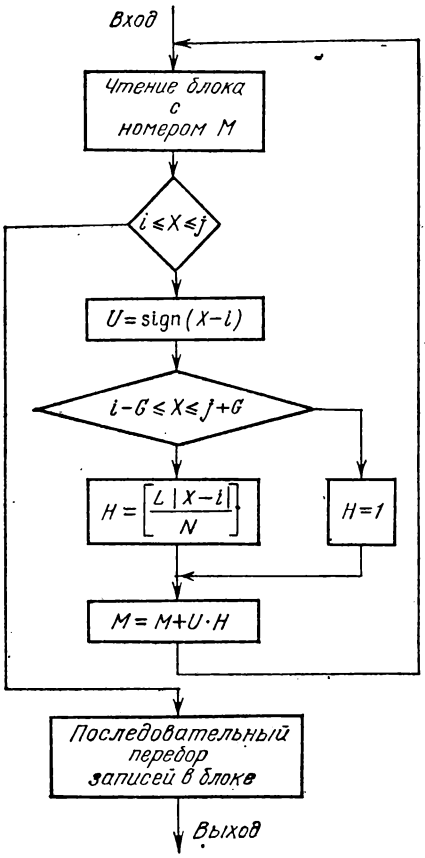


Рис. 4.2 Блок-схема алгоритма блочного поиска в последовательном файле

Величина перехода $H = [L|X - i|/N]$ зависит от удаления $|X - i|$ до искомой записи и среднего размера записи в файле L/N. Функция $U = \text{sign}(X - i)$ определяет направление перехода, а параметр G — размер окрестности искомой записи, при попадании в которую величина перехода устанавливается равной 1. Когда местонахождение записи локализовано с точностью до блока ($i \leqslant X \leqslant j$), применяется последовательный просмотр всех записей этого блока. Метод блочного поиска имеет максимальную эффективность при фиксированной длине записи в файле.

Для простоты изложения на блок-схеме не отражены меры по защите алгоритма от возможных зацикливаний или выходов в момент перехода за пределы файла. Возможность появления таких

аномалий понятна, а реализация защиты от них достаточно проста.

Применение описанного алгоритма поиска стало возможным в связи с перемещением последовательных файлов на период обработки в память с прямым доступом (в пул). Этот алгоритм поиска позволил реализовать прямой способ доступа к записям последовательного файла и сократить время поиска записи за счет уменьшения числа обменов с внешней памятью.

Для поиска записи в индексно-последовательном файле используется индекс — таблица, с которой связан алгоритм поиска. Этот алгоритм получает на входе значение ключа искомой записи и выдает информацию для быстрого ее обнаружения. Индекс создается программами индексно-последовательного метода доступа в виде многоуровневого дерева (рис. 4.3). Структура его отличается от известной структуры индекса, принятой в OS/360 [27] из-за несовпадения структур записи данных на физических носителях.

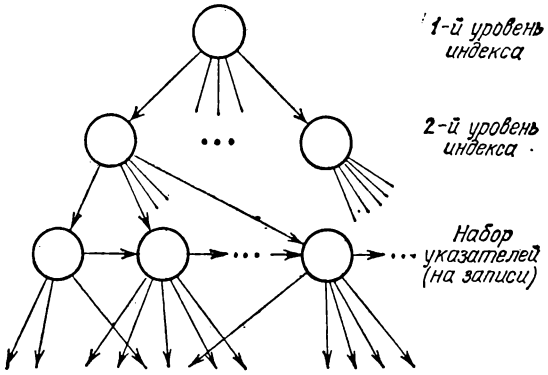


Рис. 4.3. Структура индексного дерева

Узлы дерева состоят из служебных записей, имеющих поле ключа и поле ссылки. Каждому узлу выделяется блок внешней памяти. Записи узла упорядочены по возрастанию значений ключей. На нижнем уровне индекса поле ссылки содержит указатель на запись файла или на блок внешней памяти, в котором она может находиться. Нижний уровень индексного дерева называется *набором указателей.*

Количество уровней дерева (K) заранее не определено и зависит от числа записей в файле (N), длины ключа записи (l), размера блока (L). Примерная взаимосвязь этих параметров такова:

$$K \sim \log_n N,$$

где $n \leqslant [L/(l+1)]$ — число ветвей, выходящих из одного узла.

Новый уровень индекса строится автоматически при заполнении блока, выделенного под корневой узел. Заполнившийся корневой узел разделяется на два узла, а над ним создается новый корневой узел. Древовидная организация индекса требует больше памяти, так как наименьший ключ каждого узла вынесен в узел более высокого уровня, зато такая организация индексов увеличивает скорость поиска записи по ключу.

Поиск записи по ключу осуществляется по нисходящим ветвям от корневого узла к набору указателей. При этом на каждом уровне дерева осуществляется поиск только в одном узле. Для уменьшения числа обменов при поиске по индексу наиболее часто используемые блоки индекса буферизуются в оперативной памяти.

Для обеспечения последовательной обработки файла используется набор указателей, т. е. нижний уровень индексного дерева. Как и при сканировании последовательного файла, в данном случае используется указатель на текущую запись, который движется по набору указателей. Для продвижения по набору указателей без обращения к верхним уровням индекса между узлами нижнего уровня имеются «горизонтальные» ссылки, которые направлены в сторону увеличения значений ключа.

Организованный таким образом индекс позволяет реализовать все возможности индексно-последовательных файлов известных систем управления данными [22, 27, 30, 37].

4.3. Управление памятью

Проблема управления памятью сводится к принятию решений о том, какие данные, на какие сроки и на каких уровнях памяти следует размещать. Оптимальная организация системы требует установления некоторого равновесия в использовании различных уровней памяти. Такое равновесие для двухуровневой системы памяти, реализованной в УПД ДИАПАК, достигается динамически — путем автоматического перемещения информационных массивов с одного уровня памяти на другой в зависимости от активности их использования. Эффективность использования двухуровневой системы памяти определяется тремя взаимосвязанными механизмами:

— алгоритмом распределения памяти;

— алгоритмом «уплотнения памяти»;

— алгоритмом замещения.

При разработке алгоритмов управления архивной памятью были использованы и адаптированы алгоритмы и методы, применяемые для динамического распределения памяти в операционных системах и широко обсуждаемые в литературе [20—23, 37—40, 57, 88, 92, 105].

4.3.1. Распределение памяти. Создание удовлетворительного алгоритма распределения внешней памяти явилось одной из основных задач при разработке УПД ДИАПАК. Различные размеры файлов и произвольные сроки их хранения, а также возможность изменения размеров файлов с течением времени значительно усложнили эту задачу.

Рассмотрим возможные способы ее решения:

1) резервирование памяти одним непрерывным участком (экстентом) по максимальному размеру файла, указанному при его определении;

2) динамическое выделение свободных блоков внешней памяти при увеличении размера файла;

3) динамическое резервирование экстентов внешней памяти различной емкости.

Первый способ ведет к существенным потерям внешней памяти по вполне очевидным причинам: во-первых, запрашиваемый объем памяти является максимальным для файла и, как правило, не будет использован полностью, во-вторых, фрагментация памяти приведет к дополнительной потере памяти, которая зависит от размера экстентов, выделенных для размещения файлов и объема распределяемой памяти.

Динамическое выделение свободных блоков внешней памяти в процессе увеличения размера файла не имеет указанных выше недостатков и в этом смысле является более предпочтительным. Но этот способ проигрывает в другом: блоки, выделенные под файл, должны быть связаны.

Связка блоков может быть реализована либо организацией списковой структуры, либо построением таблицы (карты памяти файла), в которой указаны адреса размещения блоков во внешней памяти и их связь [92].

Недостатком списковой организации является слабая надежность хранения: при потере одного из блоков может быть потеряна вся информация файла. Кроме того, списковая организация позволяет реализовать только последовательные способы доступа к записям файла.

Использование карты памяти файла для решения задачи распределения памяти может привести к дополнительным затратам оперативной памяти. При блочном распределении карты памяти файлов будут иметь большие размеры, т. е. существенно увеличится объем служебной информации, что особенно будет заметно при одновременной обработке на ЭВМ большого числа файлов (например, при широком использовании диалогового режима).

Динамическое разервирование экстентов различной емкости по мере поступления данных в файл также предполагает использование карты памяти файла, но в данном случае она может иметь

ограниченные размеры, если установить максимально допустимое число экстентов, выделяемых под один файл. Таким образом, могут быть сохранены и использованы все преимущества, связанные с использованием карты памяти файла, при незначительном увеличении нагрузки на внешнюю и оперативную память ЭВМ. Потери памяти, связанные с резервированием экстентов, могут быть устранены, если после завершения очередной обработки файла неиспользованную часть последнего экстента возвращать, сохранив за файлом лишь действительно используемую память. Не исключена потеря памяти из-за фрагментации, но в данном случае могут быть предприняты специальные способы борьбы с ней, вплоть до принудительного влияния на распределение длин выделяемых экстентов.

Ввиду того, что последний из рассмотренных способов распределения памяти обладает большой гибкостью, он взят за основу алгоритма распределения памяти в УПД ДИАПАК. Напомним, что пользователь работает с данными на логическом уровне, т. е. явных заявок на ресурсы памяти от задачи не поступает. Это дает возможность проверки разных алгоритмов с целью поиска оптимального.

Система допускает размещение файла на четырех экстентах памяти. Выделение очередного из них осуществляется динамически. Размеры экстентов подбираются в зависимости от типа носителя, уровня памяти, размера файла и текущего распределения памяти. Первоначально выбранные размеры экстентов могут быть изменены в последующем. Во-первых, это связано с тем, что после завершения обработки файла система может освободить неиспользованную часть памяти, выделенной под файл, а во-вторых, на запрос очередного экстента требуемого размера может быть получен отказ, так как свободного экстента памяти не оказалось. В таком случае могут быть запрошены два или более экстентов меньшего размера, суммарно превышающих требуемый объем.

В заявке, адресованной модулю распределения архивного пространства, кроме основных параметров (требуемая емкость памяти, имя района/пула, в котором необходимо осуществить поиск), могут быть указаны некоторые дополнительные сведения, способствующие более рациональному выбору экстентов в имеющейся свободной памяти. Эти сведения содержат:

— список томов, на которых предпочтительнее получить запрашиваемый объем памяти;

— список экстентов, выделенных файлу, в непосредственной близости с которыми необходимо попытаться найти смежные свободные участки памяти;

— разрешение на использование нескольких экстентов малого размера.

Кроме того, в заявке может быть указан способ выбора конкретного экстента, если среди имеющихся можно сделать неоднозначный выбор. Такими способами могут быть:

— «наилучшее соответствие» $\left(\min_i [(L_i - L) \geqslant 0]\right)$;

— «максимальный подходящий» $\left(\max_i [(L_i - L) \geqslant 0]\right)$;

— «первый подходящий».

В первых двух способах проводится анализ всех свободных экстентов, в последнем используется указатель на начало поиска, который циклически сдвигается по полю внешней памяти.

Возможность задания различного рода рекомендаций в заявке на память позволяет настроить систему на более эффективный алгоритм распределения памяти. Это может быть алгоритм:

— снижающий потери памяти из-за фрагментации путем укрупнения экстентов, выделенных под файл;

— сокращающий время доступа к данным путем размещения данных одного файла в непосредственной близости друг от друга;

— уменьшающий число постановок томов при переписи файлов из пула в район путем поиска для них максимального экстента и «сброса» в него совокупности файлов.

4.3.2. Уплотнение памяти и алгоритм замещения. Алгоритмы уплотнения памяти предназначены для уменьшения потерь памяти из-за фрагментации. Отметим, что практически все способы борьбы с фрагментацией направлены на увеличение размеров как свободных, так и занятых участков памяти. В одних случаях это делается явно, путем переписи информации с одного места на другое, и эффект от уплотнения получается незамедлительно. В других случаях укрупнение свободных и занятых участков памяти выполняется в ходе функционирования системы при наступлении подходящей ситуации. В УПД ДИАПАК реализованы оба способа уплотнения памяти.

Уплотнение памяти путем перемещения файлов вплотную друг к другу не является основным способом и инициируется администратором данных в исключительных случаях. Этот способ имеет смысл применять только для районов, в которых информационный поток имеет малую скорость (например, в программных районах).

Для уплотнения памяти в ходе функционирования системы используются специальные приемы, которые заключаются в изменении распределения длин экстентов в зависимости от текущего распределения внешней памяти. Один из таких приемов уже рассмотрен в алгоритме распределения памяти, когда увеличение занятых участков памяти достигается путем использования свободных экстентов малого размера и смежных экстентов. Другой прием укруп-

нения занятых участков памяти заключается в объединении нескольких экстентов в один экстент. Такая сборка выполняется при переписи файла из пула в район. Если файл вновь когда-нибудь будет переписываться в пул, то пространство ему будет предоставлено по возможности из одного экстента, в худшем случае из двух-трех экстентов.

Реализованный алгоритм уплотнения памяти прост, эффективен и не требует применения специальной дорогостоящей реорганизации уже распределенной памяти.

Динамическое распределение памяти в сочетании с изменением распределения длин выделяемых экстентов позволило снизить потери памяти в пулах до 5%. Отметим, что при выделении памяти под файл одним экстентом величиной в размер файла, потери памяти составили бы 15 — 20%. Эти данные получены на конкретных информационных потоках.

Необходимое равновесие в размещении информации по уровням памяти обеспечивается алгоритмом замещения. Алгоритм замещения в рассматриваемой системе инициирует переписи файлов с уровня на уровень без дополнительных указаний пользователя или его программы. При этом более быстрая стационарная память пулов используется для хранения наиболее часто используемой в данный период информации. Длительное хранение отработавшей информации осуществляется в районах на съемных магнитных дисках и магнитных лентах.

Пространство для размещения новых файлов всегда выделяется в пуле. Существующий файл становится доступным задаче только тогда, когда он находится в пуле. Если файл в пуле отсутствует, решение задачи приостанавливается, инициируется служебная задача переписи файла из района в пул (процесс «вталкивания»), после завершения которой продолжается решение задачи пользователя. Для сокращения времени простоя процесс «вталкивания» инициируется при открытии файла, а прерывание задачи наступает при непосредственном обращении к файлу. Если в результате обработки файла произошла его модификация, то формируется указание о необходимости переписи его обратно в район. Непосредственная перепись файла из пула в район (процесс «выталкивания») инициируется или по указанию оператора ЭВМ, или автоматически по достижении некоторого насыщения пула модифицированными файлами. «Выталкивание» файла необходимо выполнять как для обеспечения сохранности, так и для своевременной подготовки пула к разгрузке. После завершения «выталкивания» копия файла сохраняется в пуле и может быть неоднократно использована.

Если для размещения очередного файла свободного пространства в пуле не оказалось, то такой пул считается переполненным.

В качестве действия системы для ликвидации переполнения может быть задано «изгнание» файлов, имеющих максимальное время «простоя» с момента последнего использования, или менее активных файлов. В последнем случае для определения активности файла используются такие сведения из метки файла, как время присутствия в пуле, число обращений к файлу за это время, время «простоя», время жизни файла и общее число обращений к нему, количество «изгнаний» его из пула и пересылок обратно в пул и т. д. Кроме того, всегда принимаются во внимание объем памяти, требуемой в пуле в момент возникновения переполнения, и за счет каких файлов эта память может быть получена наилучшим способом.

В системе предусмотрены меры для выявления приближающегося переполнения, предотвращения переполнения, работы с переполнением и выхода из него.

Выявление приближающегося переполнения осуществляется с целью выдачи сигнала для запуска процессов предотвращения переполнения: «выталкивание» файлов, разгрузка пула. Сигнал, вырабатываемый на этой стадии, может быть выдан либо автоматически до наступления переполнения или при непосредственном переполнении, либо с пульта оператора ЭВМ (по приказу). В системе имеется возможность регулировки момента установки предупреждающих сигналов. Первоначальную регулировку выполняет администратор данных. В ходе функционирования системы регулировка выполняется автоматически в моменты возникновения реального переполнения.

С возникновением сигнала приближения переполнения система переходит в стадию его предотвращения. Для предотвращения переполнения используется разгрузка пула. Разгрузка пула осуществляется путем переписи файлов в районы и «изгнания» файлов из пула.

Если предотвратить переполнение не удалось, система входит в режим работы с переполнением. В этом случае задача пользователя закрывается до завершения разгрузки пула, после чего, если запрос на память удовлетворен, система выходит из режима работы с переполнением и продолжает решение задачи.

ГЛАВА 5

ОБЕСПЕЧЕНИЕ НАДЕЖНОСТИ СИСТЕМЫ УПД ДИАПАК

5.1. Норма надежности системы

Подход к решению проблем надежности системы УПД ДИАПАК объединяет методы двух известных подходов: предупреждения ошибок и обеспечения устойчивости к ошибкам и отрицательному воздействию окружающей среды (к внутренним и внешним возмущениям). Из всех методов, реализующих указанные подходы, выбраны те, которые зарекомендовали себя в практике программирования (в частности, приемы и методы, использованные в системах УПД-6 и УПД-ЕС) и при создании ответственных технических систем.

Разнообразие видов искажения информации и процессов, протекающих в системе, обусловленное функционированием ее в рамках многомашинного вычислительного комплекса, потребовало разработки ряда новых методов, направленных на предупреждение этих искажений и оперативное восстановление информации, а также некоторых процессов.

К числу этих методов, которые будут описаны ниже, относятся следующие:

— создание схем процессов и алгоритмов с ограниченными последствиями отказов;

— обнаружение внешних и внутренних возмущений;

— обеспечение сохранности служебной информации и файлов;

— оперативное и автономное восстановление служебной информации и ряда процессов системы;

— восстановление файлов.

Реализация указанного подхода осуществляется по аналогии с методикой обеспечения надежности технических систем, согласно которой при проектировании закладывается требуемая надежность, при изготовлении обеспечивается запроектированная надежность, при эксплуатации поддерживается достигнутая надежность [77, 83, 106].

При создании любой системы управления данными важным вопросом является обоснование нормы надежности, т. е. задание целесообразных или хотя бы оправданных количественных требований по надежности [107]. Последствия занижения нормы непредсказуемы и часто разрушительны [72], чрезмерное завышение приводит к увеличению затрат на разработку системы, а также к снижению эффективности ее функционирования.

На выбор нормы надежности влияют характеристики взаимодействующих с системой задач и операционная обстановка, в которой предполагается функционирование системы.

5.1.1. Характеристики задач. К числу этих характеристик в первую очередь относятся:

— календарное время (Тк) расчета (по соответствующей программе или комплексу);

— процессорное время (Тцп) расчета;

— время получения (Тп) файла, используемого при проведении расчета;

— объем и срок хранения файлов.

Тк определяет длительность активного использования файлов при решении задачи, Тцп должно учитываться при задании среднего времени безотказной работы системы, а Тп фактически определяет время восстановления файла в случае его искажения или потери.

Объемы и сроки хранения файлов должны учитываться при выборе методов обеспечения сохранности файлов и оценке внешней памяти, необходимой для реализации этих методов.

Следует подчеркнуть, что рассмотрение только характеристик задач еще не отражает действительную «картину» вычислительного эксперимента. Во-первых, всегда имеет место последовательность решения большого количества вариантов задач с помощью программ и комплексов в определенном сочетании и с определенной кратностью. При этом результаты одной программы (или комплекса) используются как входные данные для другой, т. е. возрастает длительность активного использования файлов и, вследствие этого, вероятность искажения или потери файлов. Продолжение прерванного расчета, как правило, сопряжено с трудностями восстановительных мероприятий, потерями времени счета и удлинением календарного времени расчета. Во-вторых, в ВЦ, как правило, проводится одновременно несколько вычислительных экспериментов. В связи с этим резко возрастают потоки информации, увеличивается число файлов, передаваемых от задачи к задаче, а также число файлов, используемых одновременно в нескольких расчетах. Кроме того, необходимо отметить, что с усовершенствованием методик решения задач (безавостность) возрастает время их непрерывного счета на ЭВМ.

С учетом специфики задач математической физики и особенностей вычислительного эксперимента, а также последствий сбоя или отказа системы УПД ДИАПАК среднее время безотказной работы этой системы должно быть не меньше процессорного времени решения указанных задач. Учет значений Тк и Тп приводит к необходимости исключения потерь файлов из-за сбоев или отказов системы.

5.1.2. Операционная обстановка. Операционную обстановку, в которой предполагается функционирование системы УПД ДИАПАК, образуют ОС ДИАПАК и многомашинный вычислительный комплекс БЭСМ-6.

ОС ДИАПАК характеризуется высокой надежностью и готовностью к работе, а также хорошими эксплуатационными показателями [108]. Поэтому УПД ДИАПАК, являясь частью ОС, должна иметь надежность и готовность не ниже, чем другие компоненты ОС. Так были установлены: среднее время безотказной работы — около 160 часов, коэффициент готовности — около 0,95.

Реализация процессов управления данными на комплексе ЭВМ приводит к необходимости принятия специальных мер по обеспечению надежности и готовности системы. В рассматриваемом комплексе все машины равноправны, на каждой из них имеется собственный экземпляр операционной системы, задачи пользователя могут решаться на любой действующей в данный момент ЭВМ, а информационная связь при работе комплекса осуществляется через общую дисковую память. Отсюда следует, что система УПД ДИАПАК должна обеспечить контролируемый доступ к файлам и служебной информации с разных ЭВМ и гарантировать их сохранность при модификации, исключать взаимное влияние систем, работающих на разных ЭВМ, различать неработоспособное состояние отдельных машин, осуществлять синхронизацию процессов управления данными, протекающих на разных ЭВМ, своевременно оповещать системы на других машинах о всех изменениях в информационной обстановке, уметь завершать процессы, начатые на какой-либо ЭВМ и прерванные в случае ее отказа, и др. Все это приводит к значительному усложнению системы. Ее отказ на одной из ЭВМ может вызвать «зависание» систем на других машинах, т. е. потери будут составлять сумму потерь для комплекса машин.

Особо следует рассмотреть такое свойство, как готовность, характеризующее количественно безотказность и восстанавливаемость системы. Выше отмечалось, что коэффициент готовности определяется частотой сбоев и отказов и длительностью неработоспособного состояния системы. На рис. 5.1 показаны периоды, составляющие это состояние.

Очевидно, что для уменьшения длительности неработоспособного состояния системы необходимо:

— улучшать средства диагностики отказа, индикации аварийного состояния, организацию обслуживания и квалификацию обслуживающего персонала с целью сокращения времени простоя и отыскания причины отказа;

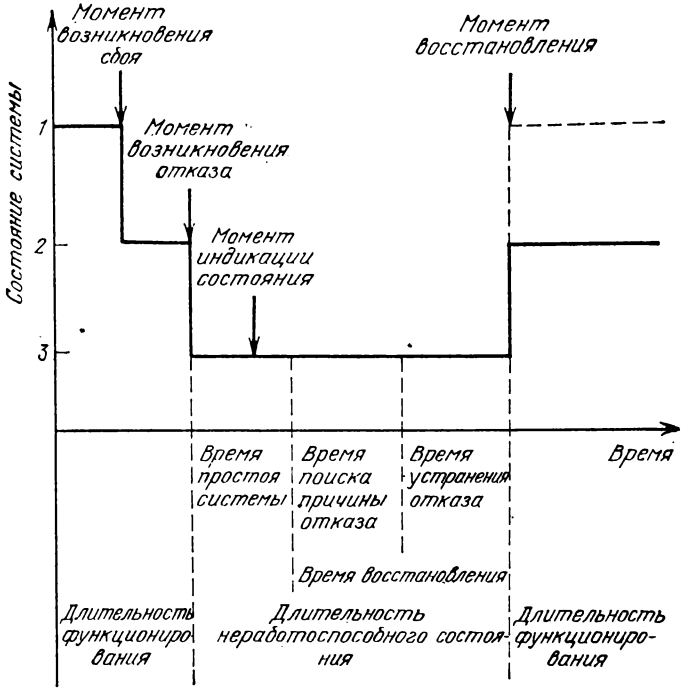


Рис. 5.1. Составляющие длительности неработоспособного состояния системы (цифрами обозначены: *1* — состояние работоспособности; *2* — состояние сбоя; *3* — состояние отказа)

— совершенствовать средства восстановления с целью сокращения времени устранения отказа.

Повышение готовности системы возможно за счет изоляции или ограничения последствий сбоя (нередко их распространение по системе приводит к ее отказу), а также за счет оперативного восстановления отказавших компонент. В ряде случаев допустимо блокирование отдельной функции или некоторой возможности системы, что снижает незначительно ее эффективность, но сохраняет ее функционирование в целом (например, временно заблоки-

ровать функцию уничтожения файла, возможность автоматического ведения поколений, автоматического уничтожения файлов по истечении срока хранения и т. п.).

Организация комплекса ЭВМ во многом определяет способ управления хранением данных. Из двух способов управления (децентрализованное и централизованное) первый предполагает создание локальных (в пределах одной ЭВМ) информационных баз. Этот путь избавляет от решения многих проблем по обеспечению надежности и готовности. Однако децентрализация жестко «привязывает» пользователя к определенной базе и нередко к определенной машине. Это противоречит основной цели организации многомашинного комплекса: концентрации вычислительных ресурсов и их гибкого распределения. Кроме того, затрудняется организация связи задач по данным. Возможные пути преодоления недостатков децентрализации приводят к существенному усложнению системы управления данными, значительному перерасходу внешней памяти и возникновению ряда новых проблем [11, 109].

Централизация управления хранением данных в рамках комплекса ЭВМ свободна от этих недостатков, но имеет свои, правда, менее серьезные недостатки. Они связаны в основном с централизацией служебной информации и появлением в ее составе объектов, уязвимых при сбоях или отказах системы. Перемещаемость этих объектов в памяти (мобильность), их частая модификация и увеличение числа связей между объектами предъявляют особые требования к обеспечению целостности (сохранности) всей служебной информации, которая содержит в каждый момент времени сведения о внешней памяти, ее логической организации [61, 62] и распределении, сведения о файлах, их местоположении, состоянии, использовании и т. п. Высокие требования предъявляются к точности и скорости восстановления служебной информации в случае ее потери или искажения.

Централизованное управление решает задачи, поставленные при организации комплекса. Недостатки этого способа могут быть преодолены с помощью соответствующих средств обеспечения надежности системы. Это будет рассмотрено в следующих параграфах данной главы.

5.2. Модель управления надежностью системы УПД ДИАПАК

Цель управления надежностью состоит в стабилизации и защите системы от отрицательного воздействия внешней среды и внутренних возмущающих факторов (программных ошибок), а также в постоянном совершенствовании методов, применяемых для

обеспечения надежности системы. На рис. 5.2 показана схема управления надежностью системы УПД ДИАПАК.

Информационные компоненты системы подвержены возмущениям внешней среды (это возмущения в виде аппаратных сбоев и отказов, ошибок ОС, пользователя, оператора и обслуживающего персонала); возможные искажения — $У$ (искажения служебной информации и файлов). Программные компоненты защищены от

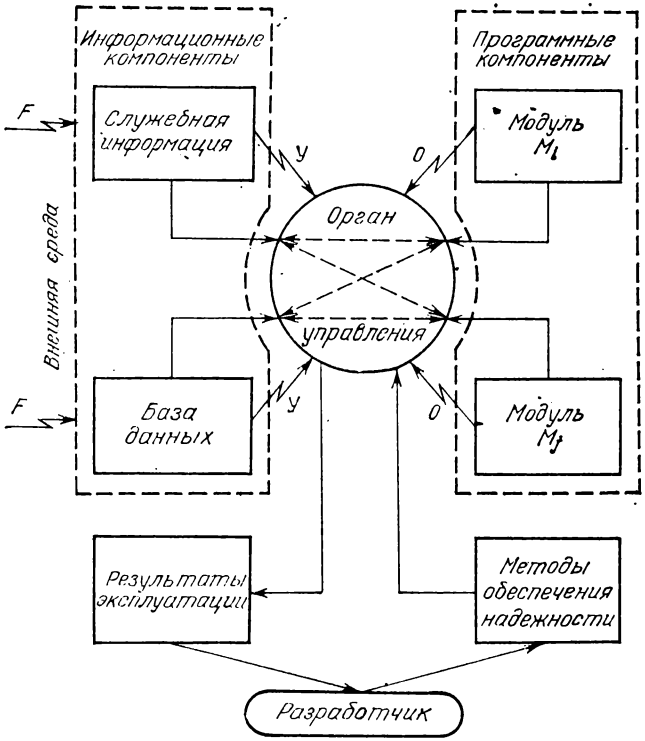


Рис. 5.2. Схема управления надежностью системы УПД ДИАПАК

внешних возмущений программно-аппаратными средствами, однако в этих компонентах не исключено наличие ошибок ($О$). Центральным элементом модели является орган управления, который контролирует взаимодействие между модулями (M_i, M_j), их работу со служебной информацией и базой данных, а также состояние последних и при необходимости осуществляет восстановление информационных компонент, ограничивает распространение

последствий ошибок или сбоев. Все это выполняется в соответствии с заранее разработанными методами обеспечения надежности.

На модели показана обратная связь, которая используется как средство последовательного улучшения управления надежностью. На основе анализа результатов эксплуатации, регистрируемых и обобщаемых органом управления и посылаемых разработчику системы, совершенствуются методы обеспечения надежности.

Орган управления включает в себя средства автоматизированного и административного управления. С одной стороны, разработка первых требует дополнительных затрат времени на проектирование, программирование и проверку, т. е. растет стоимость этих этапов. С другой стороны, надежная система имеет меньшее число отказов, а способность обнаруживать ошибки и сбои, ограничивать их последствия, диагностировать искажения служебной информации или файлов и в ряде случаев автоматически восстанавливать служебную информацию уменьшает время простоя, облегчает работу администратора системы. В целом, снижаются эксплуатационные расходы. Это подтверждается известной кривой [77], показанной на рис. 5.3 и отражающей рост стоимости проектирования и изготовления с увеличением надежности и уменьшение стоимости эксплуатации.

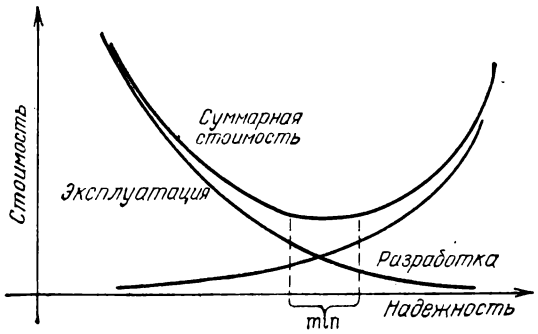


Рис. 5.3. Зависимость стоимости от надежности

Из рисунка и приведенных рассуждений следует существование интервала надежности, при котором суммарная стоимость разработки и эксплуатации минимальна.

Сложнее соотношение между эффективностью и надежностью, поскольку необходимы экономические показатели этих характеристик, которые пока отсутствуют в программировании.

5.3. Проектирование и обеспечение надежности системы

Кроме анализа класса задач, взаимодействующих с системой, и источников возмущений, рассмотренных в гл. 2, в процессе проектирования осуществляется:

— упрощение системы;

— организация взаимодействия с пользователем;

— составление имитационных схем исполнения уязвимых процессов и алгоритмов;

— введение информационной и программной избыточностей;

— определение ресурсов, необходимых для реализации выбранных методов обеспечения надежности.

5.3.1. Упрощение системы. Упрощение системы — один из трудно реализуемых методов, направленных на уменьшение ее сложности. Трудность состоит в том, что упрощение системы дает ощутимый эффект лишь при значительном уменьшении числа компонент и связей между ними. Однако сильное упрощение не позволяет обеспечить требуемую пропускную способность, требуемое время ответа, уложиться в допустимый предел памяти и т. п. Таким образом, приходится решать задачу о минимально необходимом числе компонент для реализации системы, удовлетворяющей заданным требованиям по эффективности. Такие задачи ставятся и решаются при создании аппаратуры [77]. В программировании упрощение системы достигается при помощи упорядочения связей и структуризации системы. Для этого привлекаются концепции независимости компонент, иерархической структуры, модульного и структурного программирования и др. [27, 40, 71, 72, 75]. Часть из них применяется на практике, другие опробованы при создании ряда систем [40, 75].

При создании системы УПД ДИАПАК задача упрощения решается в постановке, характерной для аппаратуры, и используются приемы, традиционно применяемые в программировании. Упрощение системы состоит в выделении оптимального по составу и функциям программного ядра, погруженного в ОС, и прикладной части, структуризации программных компонент и стандартизации интерфейса как между модулями ядра или прикладной части, так и между ядром и ОС, структуризации служебной информации.

Необходимость разделения программных компонент на ядро и прикладную часть обусловлена различиями в сложности реализации программ на системном и прикладном уровнях. Проектирование и создание модулей ядра (системных модулей) требуют высокой квалификации от разработчиков в связи с ограничениями по объему модуля, времени его исполнения, использованию ресур-

сов и т. п. Ядро образуют модули, реализующие основные процессы системы и требующие органичного сопряжения с ОС для получения доступа к ее ресурсам и сохранения порядка оформления и прохождения задачи пользователя через ВС. Прикладная часть представлена в виде подсистем определенного назначения.

Структуризация ядра (или некоторой прикладной подсистемы) заключается в разбиении его (или ее) на иерархически упорядоченные уровни. По горизонтали упорядоченность производится на основе приоритетов, присваиваемых процессам, по вертикали — вложенностью реализующих процесс модулей. На одном уровне модули независимы друг от друга. Связь между модулями одного процесса, находящимися на разных уровнях, осуществляется через стандартный интерфейс по правилам вызывающей и вызываемой программ [27, 110]. В целях экономного расхода памяти ОС для хранения модулей нижний уровень иерархии образован модулями, «работающими» с определенными объектами служебной информации (терминальные модули). Ряд процессов, имеющих общую начальную часть, объединены управляющим модулем. Они начинаются и заканчиваются в этом модуле. Сопряжение между ядром и ОС осуществляет монитор, которому передаются все заказы, инициированные пользователем, оператором или модулями ОС. Для облегчения анализа причины отказа в ядре интерфейсом обеспечивается запоминание информации об исполняемом в данный момент процессе и «работающем» модуле, единообразие представления заказов к модулям и регистрация обращений к общим блокам ядра, расположенным в мониторе.

Преимущественная каталогизация файлов, двухуровневая организация внешней памяти [62] и централизация размещения файлов на внешней памяти увеличивают объем служебной информации, необходимой для реализации этих принципов, и число связей между ее объектами. Все это усложняет переход от логического представления данных в памяти к физическому и соответствующие процессы и алгоритмы. Структуризация служебной информации ставит своей целью компактную организацию ее объектов, упрощение связей между ними и способов их адресации. Служебная информация организуется в оперативной памяти ОС для данной ЭВМ и на резидентном томе (РТ).

В оперативной памяти размещаются управляющие шкалы, обеспечивающие связь ядра с ОС и исполнение процессов системы, и таблица файлов, открытых на данной ЭВМ (ТОФ). Структурным элементом последней является блок управления файлом, где хранятся основные атрибуты файла, необходимые при его обработке, и сведения о местоположении файла на внешней памяти. Для каждой задачи, получившей доступ к файлам, блоки образуют двунаправленный список. Компактность организации блока обеспечи-

вается битовым представлением ряда атрибутов файла и заменой символических имен района, пула [62], файла физическими именами, упрощающими адресацию. Адрес блока в таблице файлов определяется по формуле (алгоритмы, реализующие поиск по формуле, будем называть *алгоритмами формульного типа*). Размер блока и всех элементов рассматриваемых ниже объектов выбран как 2^k, что упрощает формулы и их программирование, сокращает представление адресов и ссылок.

Структурным элементом служебной информации на РТ является идентифицируемый объект с фиксированным или перемещаемым местоположением. Каждый объект занимает целое число зон. У перемещаемого объекта обязательно фиксируется начальная зона. Координаты объектов (образованные парами: символическое имя — местоположение) хранятся в метке РТ. Только согласно этим координатам осуществляется поиск объектов. Из основных объектов следует выделить:

— таблицу распределения ресурсов РТ;

— каталог файлов;

— таблицу распределения пространства под файлы (пространство — это логическое представление внешней памяти, которую образуют тома, переданные системе и в последующем называемые *архивными*).

Таблица распределения ресурсов РТ регулирует размещение объектов (и их элементов) и представляет собой битовую матрицу, в которой нулевое значение бита указывает на то, что ресурс занят, единичное — ресурс свободен. Принятая структура упрощает алгоритм запроса и освобождения ресурсов — он сводится к формульному типу.

Каталог файлов имеет древовидную структуру, динамически наращиваемую до четырех уровней, достаточных для каталогизации нескольких десятков тысяч файлов только в пределах одного РТ. Динамика развития каталога файлов без внешнего вмешательства обеспечивается за счет перемещаемости элементов каталога. Однако это усложняет реализацию процессов, «работающих» с каталогом, и удлиняет цепочку преобразований его элементов в условиях многомашинного комплекса. Конфликт между противоречивыми требованиями сглажен разделением каталога файлов на перемещаемую и неперемещаемую части. Последнюю образуют корень дерева (вершина каталога) и листья (метки файлов (МФ)) [37]. При этом сохранена динамика развития каталога. Его структура и способ упорядочения информации показаны на рис. 5.4.

Таблица распределения пространства под файлы образована тремя объектами:

— каталогом районов и пулов;

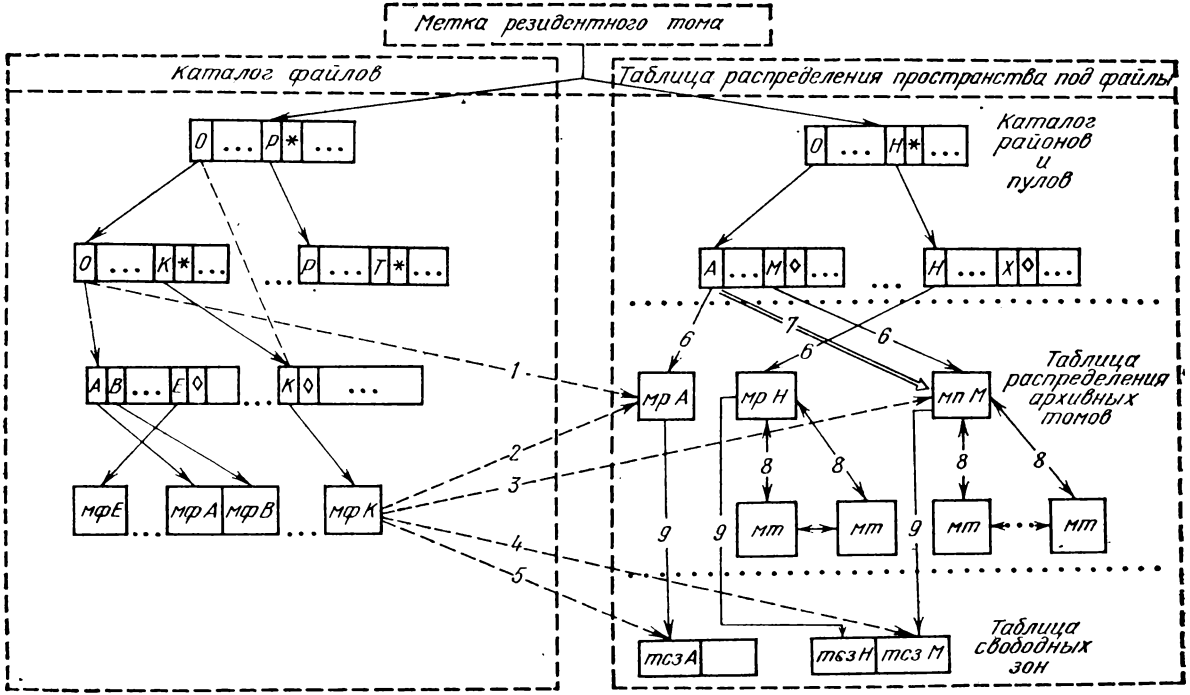


Рис. 5.4. Структура основных объектов резидентного тома и связи между ними

— таблицей распределения архивных томов;

— совокупностью таблиц свободных зон районов и пулов.

Каталог районов имеет древовидную структуру, динамически наращиваемую до двух уровней, достаточных для организации нескольких тысяч районов и пулов. Местоположение вершины каталога зафиксировано, следующий уровень является перемещаемым. Листья дерева — метки районов и пулов (МР и МП) — организуются в таблицу распределения архивных томов, которая имеет фиксированные размер и положение на РТ. Элементом этой таблицы является метка тома (МТ), зарегистрированного в системе. Определение адреса МТ осуществляется алгоритмом формульного типа. Метка головного тома района или пула [62] одновременно является меткой этого района или пула. Такое совмещение уменьшает число связей в логической структуре внешней памяти. Физическое имя района или пула совпадает с учётным номером головного тома, т. е. значение поля элемента является и ссылкой на другой элемент, что приводит к унификации формул определения адресов элементов.

Таблица свободных зон района или пула (ТСЗ) представляет битовую матрицу, в которой нулевое значение бита указывает на то, что зона занята, единичное — зона свободна. Эти таблицы также могут перемещаться по РТ. Структура таблицы распределения пространства под файлы и связи между ее объектами показаны на рис. 5.4. На рисунке изображены связи (помеченные цифрами) между элементами каталога файлов (КФ) и таблицы распределения пространства под файлы (ТРПФ), прямоугольником отмечены зоны уровней КФ и каталога районов и пулов (КРП), а также таблиц свободных зон. Метки обозначены квадратом. Элементом верхних уровней КФ и КРП является информационная группа. Символы *, ◇ обозначают конец групп в зоне.

Подробное изложение структуры служебной информации необходимо для понимания рассматриваемых ниже процессов управления данными.

5.3.2. Организация взаимодействия с пользователем. В разрабатываемых системах управления данными язык, предоставляемый пользователю, должен соответствовать обеспечиваемому операционной системой интерфейсу между пользователем и ЭВМ. Очень часто за стандарт принимается язык какой-либо известной системы (например, OS/360), который адаптируется к условиям, предоставляемым другой операционной системой. В результате он становится непохожим на те средства, которые обеспечивались операционной системой. Это создает трудности в освоении языка управления данными и сопряжении с языками программирования, приводит к изменению способа оформления пакета для ВС и в конечном счете является причиной многих ошибок пользователя.

Организация взаимодействия с пользователем преследует две цели: минимизацию ошибок пользователя и исключение влияния их на работу системы.

Первая цель достигается упрощением языка взаимодействия с системой, стандартизацией форматов сообщений, принимаемых и выдаваемых пользователю, и обеспечением его удобными сервисными средствами. Пользователь имеет дело с логическими объектами: районами, файлами и записями. Привычные для пользователя способы описания и доступа к данным нашли отражение в языке управления районами и файлами и операциях над записями. Формат процедур языка единообразен как при статическом, так и при динамическом задании их [62]. В целом, для большинства пользователей обеспечено соответствие интерфейса с навыками их работы и требованиями их задач. Сервисные средства, как отмечалось ранее, объединены и представлены диалоговой подсистемой УНИПАК. Язык подсистемы обеспечивает взаимодействие с ней в терминах УПД ДИАПАК. Команды языка (директивы) инициируют задание или диалог, в ходе которого уточняется задание. Активной стороной при диалоге является подсистема, при этом предусматривается «помощь» пользователю — выдача подсказок.

Вторая цель достигается контролем операций управления районами, файлами и обработки файлов с момента приема операций и до начала инициируемых ими процессов. Различаются два типа ошибок, допускаемых пользователем в этих операциях, а также при «общении» с подсистемой УНИПАК:

— синтаксические ошибки, являющиеся нарушением правил языка взаимодействия;

— логические ошибки, связанные с обращением к несуществующим районам, файлам и записям, нарушением правил доступа к ним (район → файл → запись), выходом за пределы района или файла и т. п.

На рис. 5.5 представлена стандартная схема обнаружения ошибок пользователя. Поскольку возможны искажения служебной информации и сбои аппаратуры, схемой предусматривается выдача сообщений о «фатальной» ошибке. Согласно схеме ошибка пользователя может быть обнаружена до начала процесса. При исполнении процесса возможны сообщения только о «фатальных» ошибках.

Приведенная схема рассчитана на обнаружение ошибок, когда пользователь общается с системой по разрешенным каналам, все другие каналы (см. рис. 2.7) должны быть закрыты для пользователя.

Взаимодействие с оператором и администратором [68] организуется аналогичным образом.

5.3.3. Имитационные схемы. При проектировании систем, устойчивых к внешним и внутренним возмущениям, необходимо определение точек уязвимости различных процессов, протекающих в системах. Методика, специально разработанная для этих целей при создании системы УПД ДИАПАК [65], основана на анализе процессов с помощью имитационных схем исполнения. Прежде

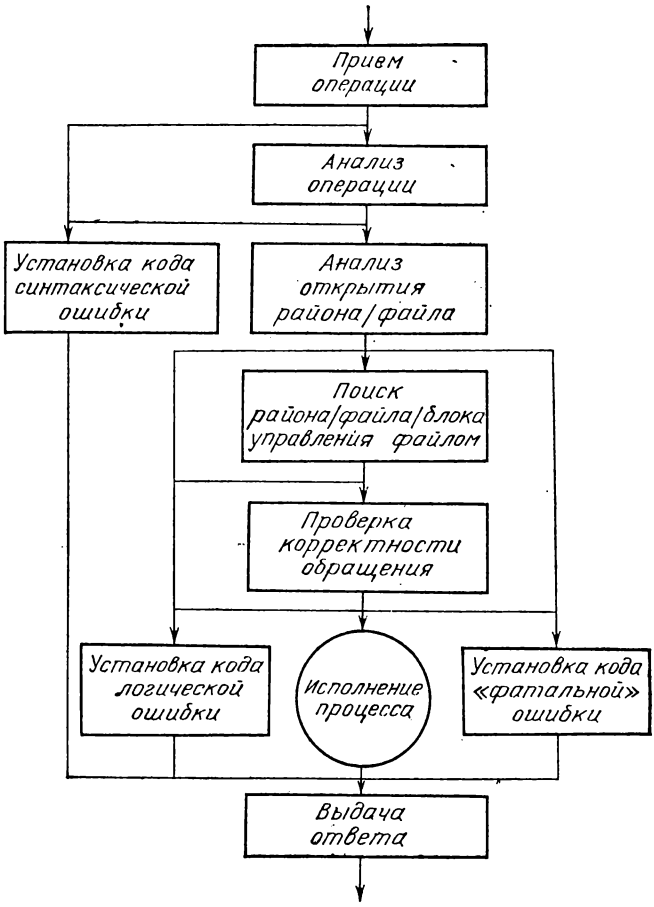


Рис. 5.5. Схема обнаружения ошибок пользователя

всего выделяются жизненно важные для функционирования системы объекты служебной информации, искажение которых может привести к искажению или потере файлов. К таким объектам относятся таблица открытых файлов, таблица распределения ресурсов РТ, каталог файлов, таблица распределения пространства под

файлы. Сохранность последовательных, индексно-последовательных и библиотечных файлов зависит также от целостности информации, описывающей структуру их хранения (заголовки записей, справочники и оглавления файлов). Эта информация размещена «внутри» самих файлов. Как отмечалось ранее, уязвимость информационных компонент повышается при их модификации. Процессы, связанные с модификацией, суть:

— определение и уничтожение района или пула;
— увеличение и уменьшение размера района или пула;
— определение и уничтожение файла;
— внесение записи в индексно-последовательный файл;
— уплотнение каталога файлов.

На рис. 5.6—5.9 изображены схемы перечисленных процессов (для иллюстрации сложности и длины цепочек преобразований служебной информации приведены схемы процессов открытия и закрытия файлов). Каждая схема отражает алгоритмы и модифицируемые ими объекты служебной информации или справочник файла, а также возможные пути «развития» процесса. Любой процесс инициируется соответствующей операцией пользователем или администратором. С момента выдачи операции до начала процесса (отмеченного на рисунках овалом) логика работы системы единообразна: прием операции, анализ операции, проверка открытия района или файла (если принята операция управления файлами или обработки файлов), поиск объекта (метки района/пула, метки файла, блока управления файлом), на который указывает операция, и анализ правомочности (пользователя или администратора) выполнения данной операции. Эти фазы работы системы были рассмотрены ранее (см. рис. 5.5). Каждый процесс завершается выдачей ответа.

В соответствии с указанными схемами для каждого процесса составляется имитационная схема исполнения в предположении возможных отказов ВС, прерывающих процесс, искажений объектов служебной информации (или справочника файла) и т. п. В качестве иллюстрации рассматривается пример составления имитационной схемы использования процесса определения файла (рис. 5.10).

Слева на рисунке приведены возможные искажения объектов служебной информации, с которыми взаимодействуют алгоритмы процесса, и отмечены точки прерывания процесса (О.1 и т. д.), справа перечислены последствия указанных возмущений. Более интересно рассмотреть алгоритм «деления» зоны. Еще до начала процесса, при поиске места в каталоге файлов для очередной информационной группы, составляется таблица, отражающая траекторию пути по дереву. В этой таблице хранятся номера зон, отмеченных на данном пути, и признаки, указывающие на необходи-

мость «деления» зон после вставки информационной группы. Если зона заполнена, то для ее «деления» требуется одна или более свободных зон в зависимости от номера уровня, где находится эта зона. Далее осуществляется перемещение части информационных групп из «делимой» зоны в свободную зону РТ, формирование

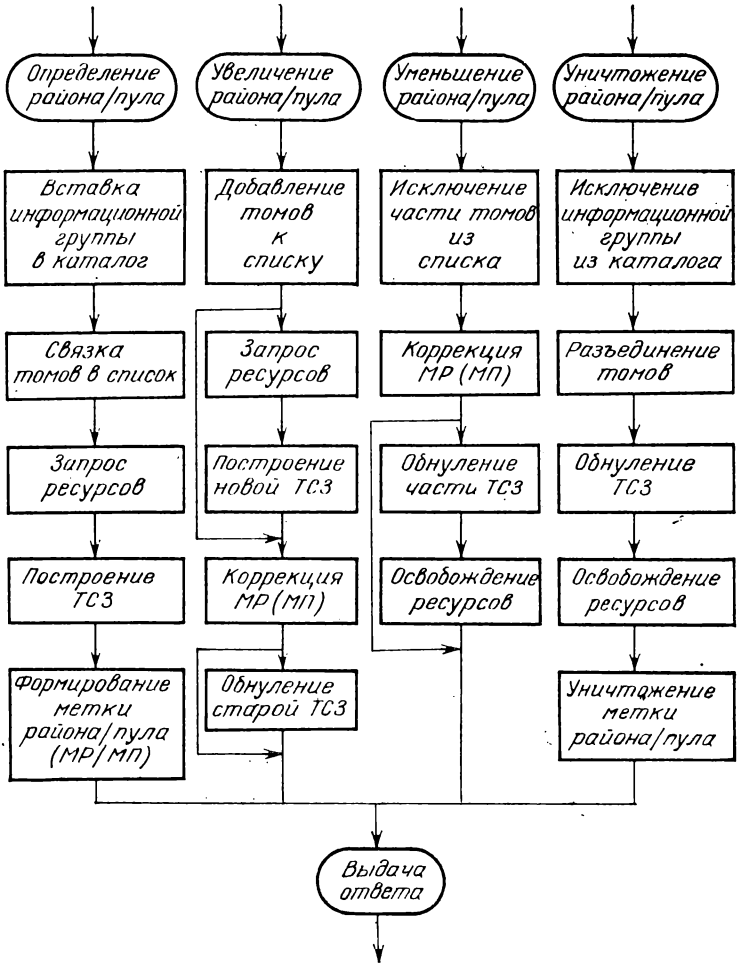


Рис. 5.6. Схемы процессов управления районами/пулами

группы, указывающей на появление новой зоны уровня, и включение этой группы в соответствующую зону предыдущего уровня. Такие действия повторяются (см. рис. 5.7, 5.10), если заполненными оказываются отмеченные в упомянутой таблице зоны.

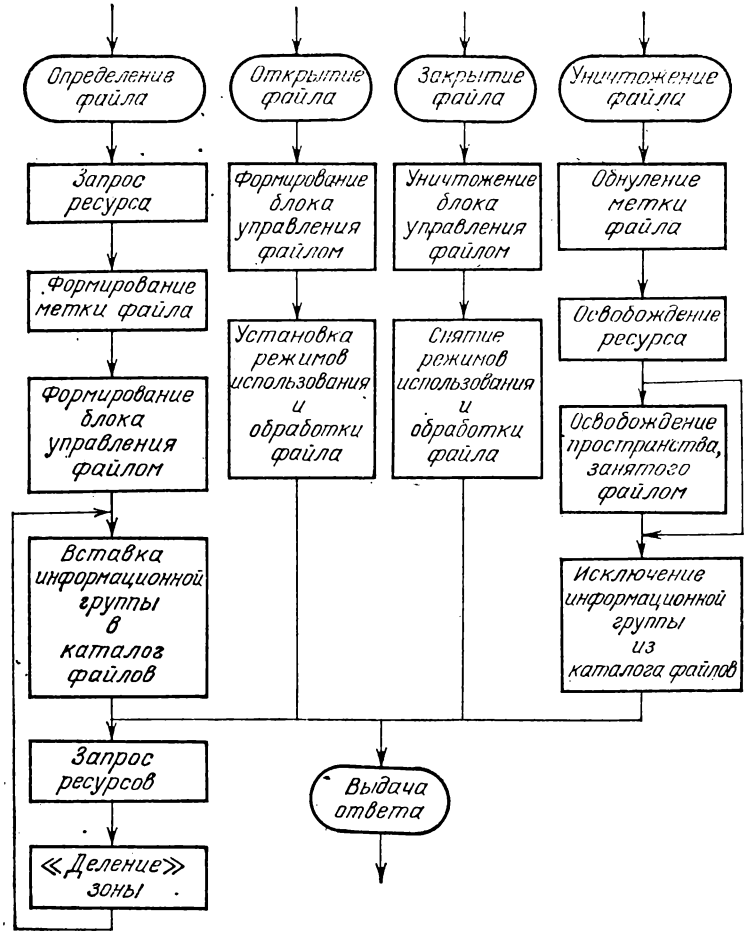


Рис. 5.7. Схема процессов управления файлами

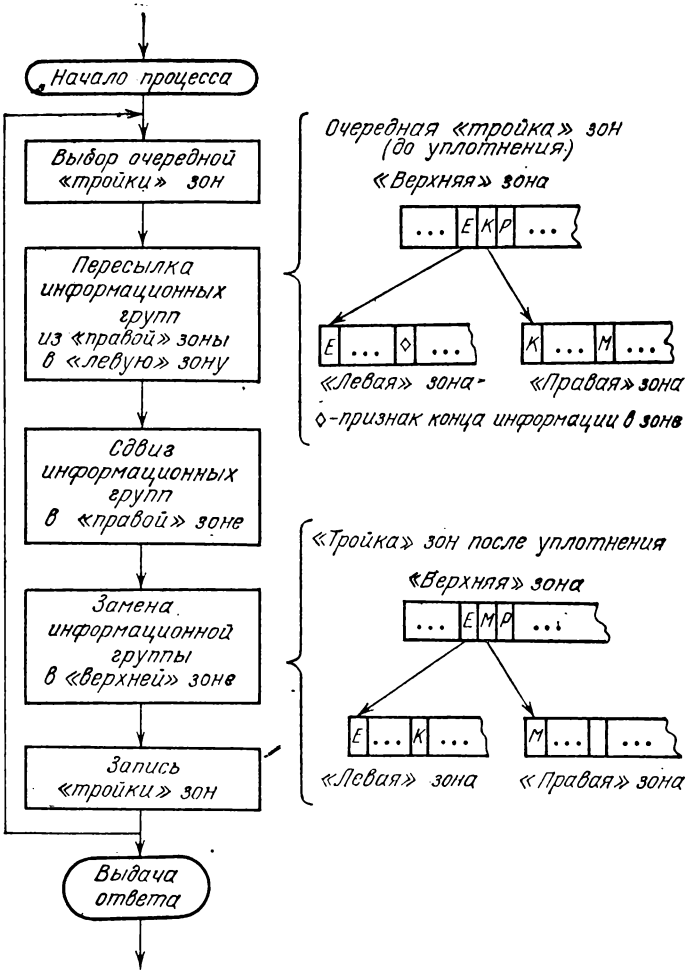


Рис. 5.8. Схема процесса уплотнения каталога файлов

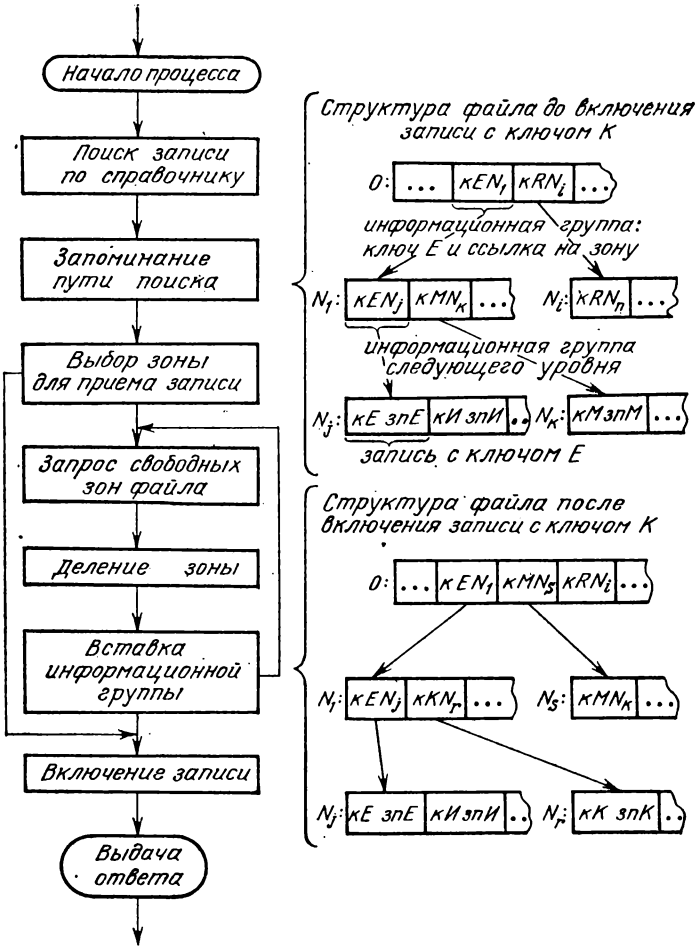


Рис. 5.9. Схема процесса включения записи в индексно-последовательный файл

Анализ алгоритма «деления» зоны показывает, что при различных последовательностях выполнения преобразований служебной информации различна его уязвимость в случае отказа. Это иллюстрируется двумя схемами данного алгоритма. Первая отражает

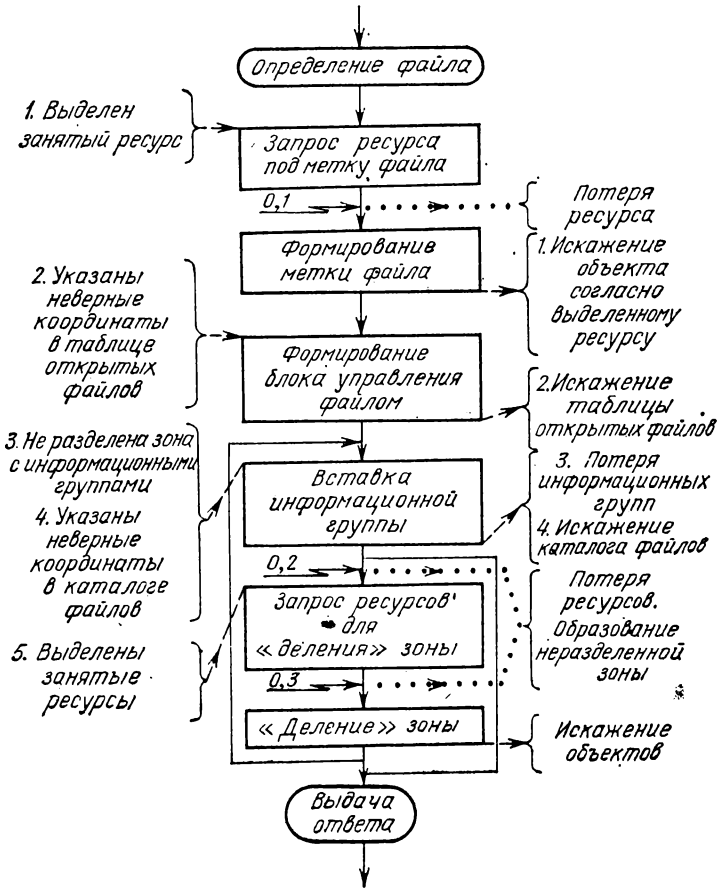


Рис. 5.10. Имитационная схема исполнения процесса определения файла

естественный порядок выполнения преобразований (рис. 5.11), вторая учитывает возможность отказа (рис. 5.12). На рисунках используются следующие обозначения: $C_1^{пр} \rightarrow C_1^{л}$ означает перемещение правой половины страницы C_1 в левую $O \rightarrow C_1^{пр}$ — очистка правой половины C_i.

При отказе в точке *1* на первой схеме произойдет потеря правой половины «делимой» зоны, поскольку информационная группа, указывающая на новую зону, не появляется на предыдущем уровне. При отказе в точке *1* на второй схеме потери информации не произойдет — в правой половине «делимой» зоны и новой зоне

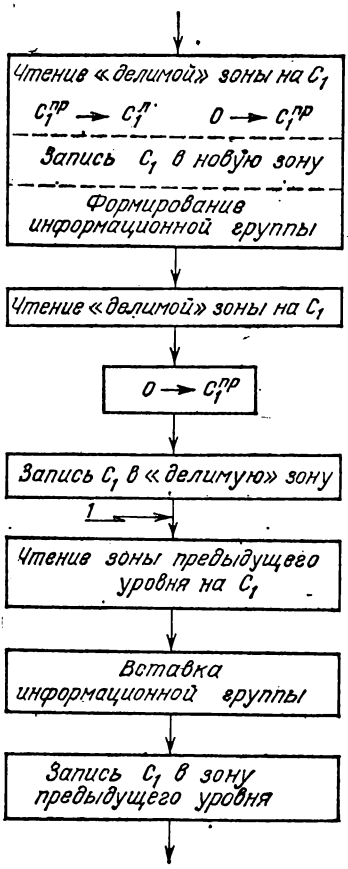


Рис. 5.11. Схема алгоритма «деления» зоны (чувствительная к отказам)

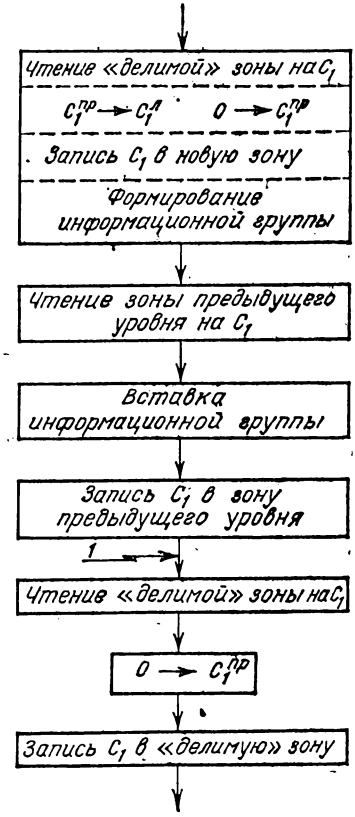


Рис. 5.12. Схема алгоритма «деления» зоны с ограниченными последствиями отказов

окажется одна и та же информация, что не повлияет на работу системы. Однако, если в неразделенную зону будет поступать очередная информационная группа, указывающая на вновь сформированную метку файла, повторится дублирование правой половины такой зоны. В каталоге файлов будет постоянно накапливаться

«мусор» и, следовательно, расходоваться пространство РТ. Поэтому при вставке группы в каталог необходимо реагировать на неразделенные зоны, что отражается в имитационной схеме (см. рис. 5.10) соответствующей записью (слева и справа).

Составленная таким образом имитационная схема определяет последовательность исполнения процесса, точки его уязвимости и возможные меры по минимизации потерь в случае его прерывания.

5.3.4. Информационная и программная избыточности. Для снижения влияния внешних и внутренних возмущений на функционирование системы вводятся и используются информационная и программная избыточности.

Информационная избыточность используется для обеспечения достоверности жизненно важных объектов служебной информации и информации, описывающей структуру хранения последовательных, индексно-последовательных и библиотечных файлов. Введение этого вида избыточности заключается в организации идентифицирующих полей внутри объектов и их элементов, заголовков записей, справочников и оглавлений файлов, дополнительных связей между элементами объектов и в структуре хранения файлов, дублировании ряда полей в различных элементах, дублировании самих объектов и справочников библиотечных файлов, копировании всего РТ и подсчете контрольных сумм. Информационная избыточность позволяет обнаружить и быстро устранить искажение объектов служебной информации, заголовков записей, справочников и оглавлений файлов.

Программная избыточность необходима для реализации средств обнаружения внешних и внутренних возмущений, индикации, диагностики и восстановления работоспособности системы.

Прежде всего следует отметить, что ОС для системы УПД ДИАПАК выступает как система «раннего обнаружения»: фиксирует некоторые виды отказов оборудования и предотвращает распространение отрицательных последствий на свои компоненты; обеспечивает защиту от ошибочных действий пользователя, оператора и администратора. Таким образом, средства ОС образуют первый уровень защиты системы УПД ДИАПАК от внешних возмущений. Второй и следующие уровни представляют собой встроенные в ядро системы средства контроля входных данных, обнаружения собственных ошибок и искажений служебной информации, структуры хранения последовательных, индексно-последовательных и библиотечных файлов, оперативного дублирования и восстановления служебной информации и, наконец, подсистему автономного контроля служебной информации (СИ) (рис. 5.13).

Средства ОС блокируют обращение задач, не взаимодействующих с системой УПД ДИАПАК, к архивным томам, обеспечивают статус системного тома для РТ, что исключает загрузку в ОП за-

дачи пользователя, в паспорте которой указан этот том, контролируют применение средств ограниченного использования, выполняют первичную проверку операций управления районами, файлами и операций ввода-вывода записей.

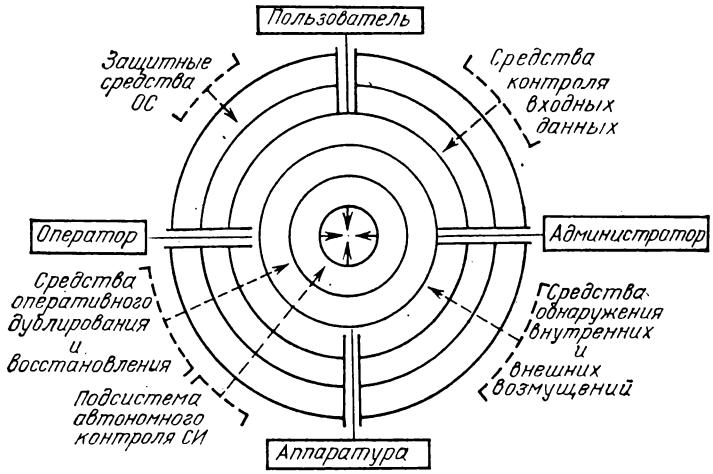


Рис. 5.13. Схема защиты системы УПД ДИАПАК от внешних и внутренних возмущений

Средства контроля входных данных (см. рис. 5.5) совместно со средствами ОС позволяют исключить влияние ошибок пользователя, оператора и администратора на систему (кроме случаев преднамеренного искажения служебной информации или базы данных со стороны последних).

Средства обнаружения собственных ошибок и искажений в служебной информации или структуре хранения файлов позволяют ограничить влияние ошибки на процесс, протекающий в системе, и распространение искажений, локализовать ошибку и этим ускорить ее поиск и исправление, своевременно оповестить обслуживающий персонал о всех обнаруженных искажениях и этим сократить время на восстановление. Данными средствами осуществляется контроль:

— входных и выходных данных между модулями ядра (аналогично и в подсистемах прикладного уровня);

— целостности объектов служебной информации, заголовков записей, справочников и оглавлений по допустимым значениям выделенных полей, контрольным суммам, упорядоченности каталога файлов, каталога районов и пулов и т. п.;

— правильности связей между объектами и их элементами в структуре хранения файлов.

Выход данных за пределы допустимых значений, искажение служебной информации и нарушение связей рассматриваются, прежде всего, как симптомы ошибки в системе. Возможность воспроизведения аварийной ситуации позволяет уточнить диагноз.

Реакция системы на аварийную ситуацию может быть различной:

— повторить операцию, выполнение которой завершилось с ошибкой (имеются в виду операции обмена с внешней памятью);

— передать управление на блоки оперативного восстановления;

— завершить задачу пользователя, при выполнении заказа которой возникла эта ситуация;

— блокировать функцию системы, при реализации которой возникает данная ситуация;

— прекратить функционирование системы (исполнение команды СТОП).

Во всех случаях на операторский терминал выдается соответствующее сообщение с указанием характера сбоя или отказа, их места и последствий для задачи пользователя. В качестве иллюстрации работы средств обнаружения ошибок и искажений служебной информации приводится фрагмент схемы процесса уничтожения файла (рис. 5.14). На рисунке обозначены: шестиугольником — начало модуля, овалом — формируемое сообщение, прямоугольником — основные функции, выполняемые модулями, квадратом — команда СТОП; выделены также точки возврата (*A*, *B*, *C*) из вызываемых модулей.

При частой модификации служебной информации, высокой мобильности ее объектов, динамическом распределении пространства под файлы и перемещении их между уровнями ВП целесообразно использование метода оперативного дублирования для обеспечения сохранности служебной информации. Этот метод позволяет ускорить восстановление служебной информации в случае ее искажения или отказа РТ. Метод заключается в создании идентичных объектов в основной и дубль-областях, организуемых на РТ или (для большей надежности) на РТ и его дубле (ДРТ). Всякая запись в РТ одновременно сопровождается записью в ДРТ. Уменьшению вероятности переноса искаженной информации из основной области в дубль способствуют анализ состояния объектов в ходе исполнения процессов и контроль в момент записи. Последний исключает запись в ДРТ страницы обмена, если изменилась информация в ней после записи в РТ (рис. 5.15; там же приведена

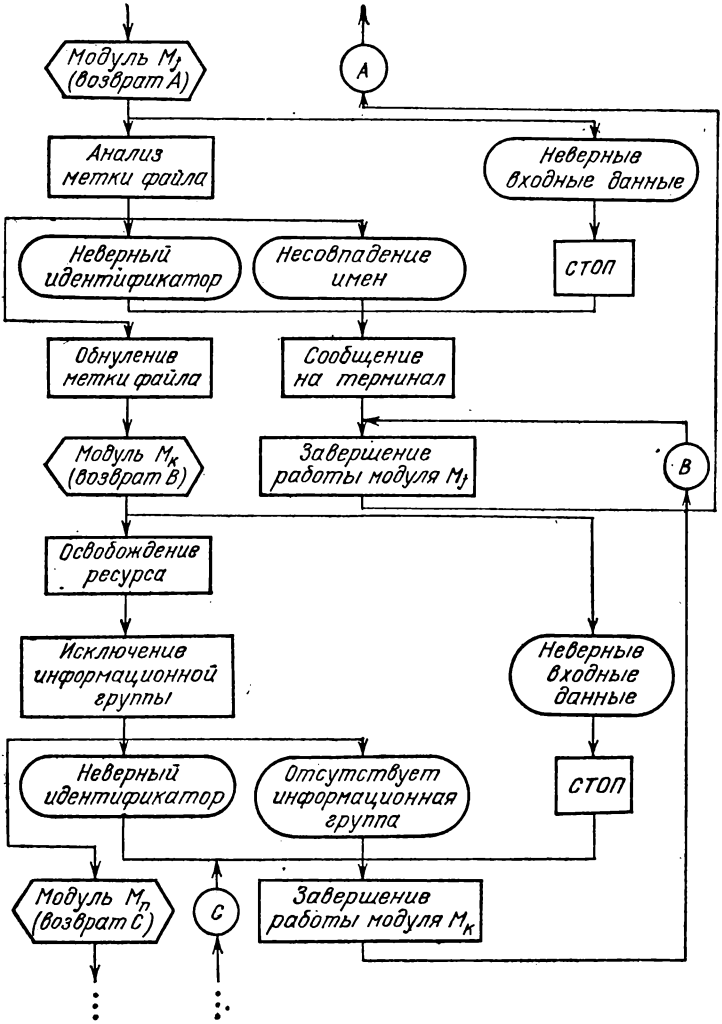


Рис. 5.14. Фрагмент схемы процесса уничтожения файла

схема восстановления зон РТ). Дублированию подлежит таблица открытых файлов для обеспечения быстрого восстановления после искажения и автоматического закрытия районов и файлов после

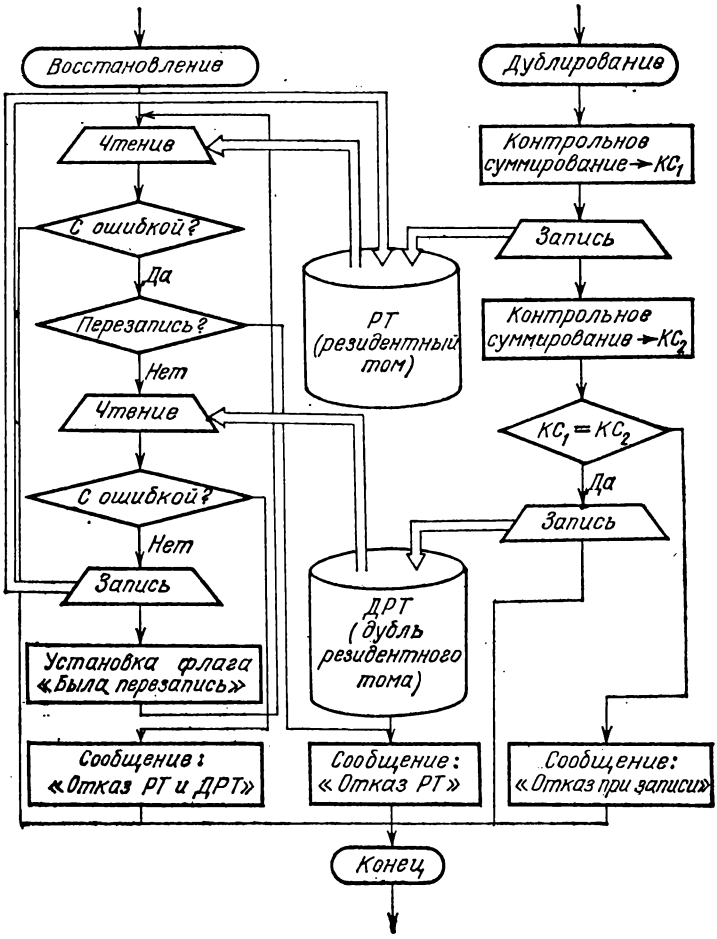


Рис. 5.15. Схема оперативного дублирования и восстановления служебной информации

перевызова ОС. Последнее необходимо для приведения в соответствие с действительным состоянием файла значений ряда полей в метке файла, регулирующих его монопольное и совместное исполь-

зование, перемещение из пула в район после модификации и фиксирующих фактически использованное пространство в файле.

Помимо оперативного дублирования объектов РТ, предусматривается его копирование для восстановления после катастрофических отказов, например повреждения РТ и ДРТ, затирания какого-либо объекта одновременно на обоих томах и т. п. Полученная копия отражает состояние РТ на определенный момент времени и подвергается тщательному анализу с помощью подсистемы автономного контроля служебной информации. Копирование целесообразно проводить не реже одного раза в сутки в подходящее для системы время. При таком регламенте обеспечивается ежесуточный контроль состояния РТ и восстановление после катастрофических отказов с меньшими потерями файлов. Число томов для копирования выбирается таким, чтобы сохранить достоверные копии, если в самые «свежие» отражается уже искаженная информация. Процесс копирования инициируется оператором по соответствующей директиве. Заказ передается монитору, который запускает данный процесс (его схема приведена на рис. 5.16). Монопольное использование РТ исключает изменение служебной информации в момент копирования.

Подсистема контроля служебной информации (КСИ) работает с томом копии, номер которого передан в ее паспорт при запуске. Она осуществляет полную проверку всех объектов РТ и связей между ними (рис. 5.17). Подсистема КСИ фиксирует следующие отклонения от нормы:

— искажение значений полей элементов;

— рассогласование связей;

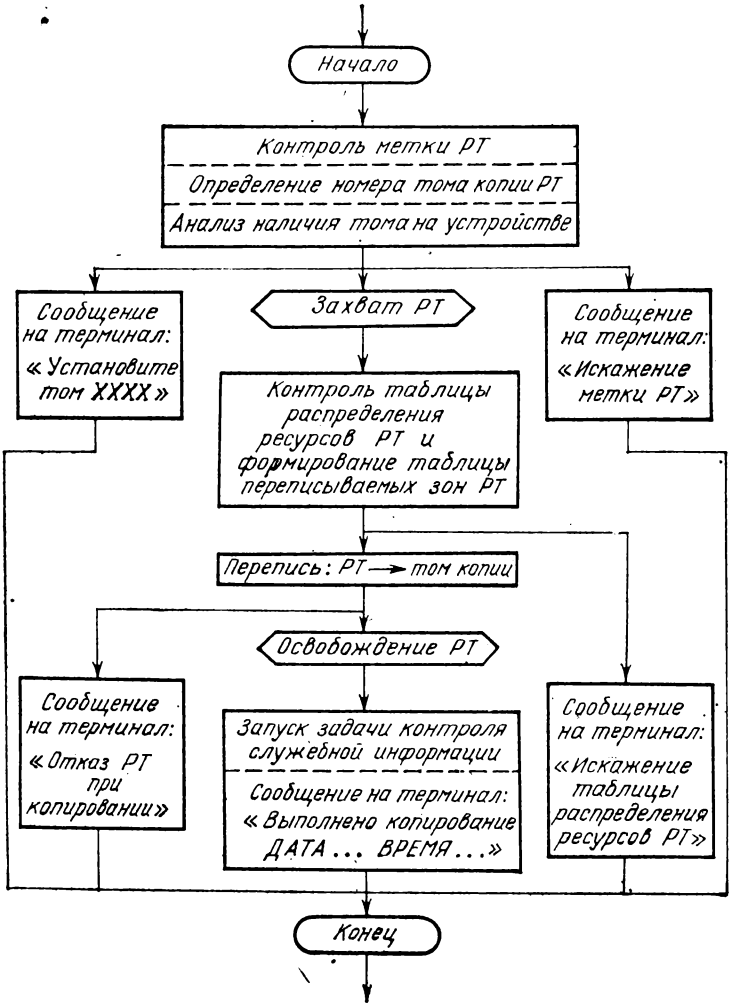
— пересечение объектов РТ (один и более ресурсов отведены двум и более объектам);

— пересечение файлов (выделенные зоны принадлежат одновременно двум и более файлам);

— потери ресурсов РТ (ресурс отмечен в таблице распределения ресурсов как занятый, но не принадлежит ни одному из объектов);

— потери пространства в районах и пулах.

Подсистема КСИ осуществляет коррекцию значений некоторых полей в метке файла при несоответствии их действительному состоянию файлов. В случае нормального завершения контроля и отсутствия серьезных искажений служебной информации подсистема изменяет в метке резидентного тома номер очередного тома копии. Фактически она является диагностической программой, которая по состоянию служебной информации на РТ определяет работоспособность системы, указывает на симптомы ошибок в последней и предупреждает о возможных отклонениях процессов системы.



Рис, 5,16, Схема копирования резидентного тома (РТ)

Обеспечение целостности файлов остается серьезной проблемой, поскольку системе неизвестна семантика данных. Пользователю предоставляются средства копирования и восстановления файлов, ведения их поколений и контроля структуры хранения. Наличие двухуровневой памяти можно рассматривать как фактор повышения сохранности файлов, поскольку в районе хранятся их ос-

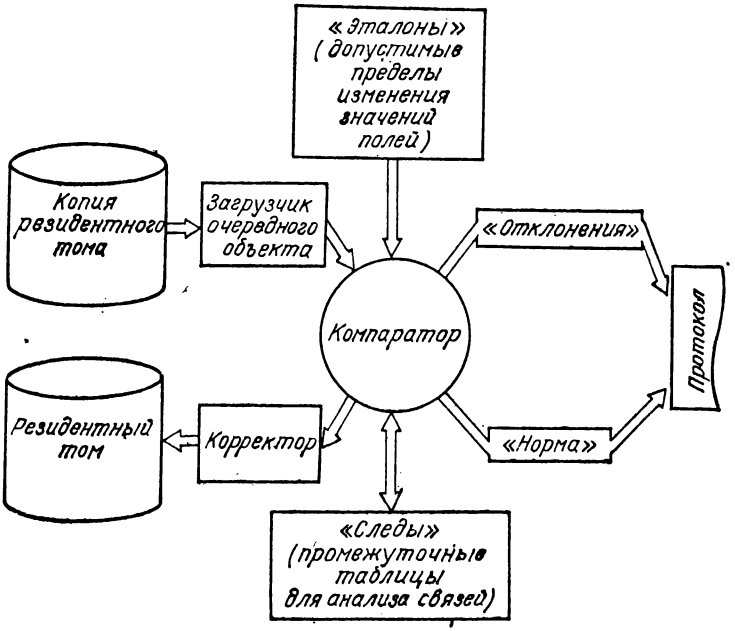


Рис. 5.17. Схема работы подсистемы контроля служебной информации

новные экземпляры, а в пуле осуществляется обработка копий. При этом перемещение модифицированного файла из пула в район является своеобразным копированием без участия пользователя.

5.4. Поддержание надежности при эксплуатации системы

В процессе промышленной эксплуатации системы основной задачей считается поддержание достигнутой надежности. Для ее выполнения предусматриваются следующие мероприятия:

— разработка инструкций по эксплуатации и восстановлению системы и данных;

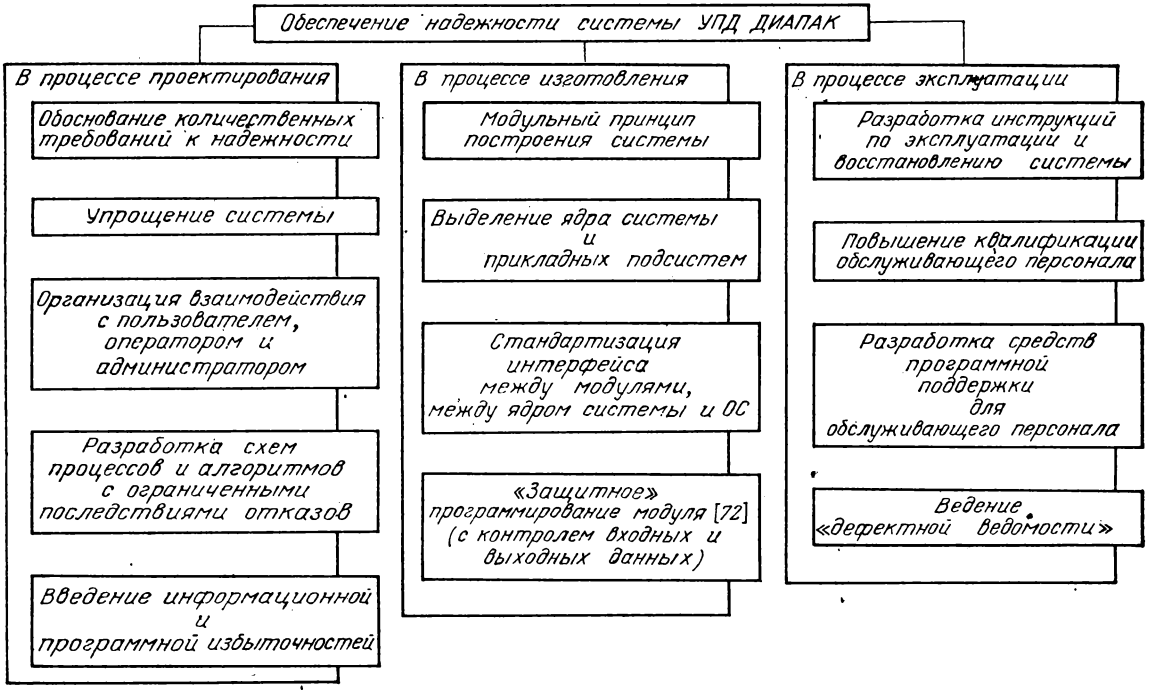


Рис. 5.18. Схема мероприятий по обеспечению надежности системы УПД ДИАПАК

— повышение квалификации обслуживающего персонала;

— разработка средств программной поддержки деятельности обслуживающего персонала;

— ведение «дефектной ведомости» для регистрации обнаруженных ошибок, сбоев и отказов системы, их причин, последствий и времени устранения.

Окончательная схема мероприятий по обеспечению надежности системы представлена на рис. 5.18.

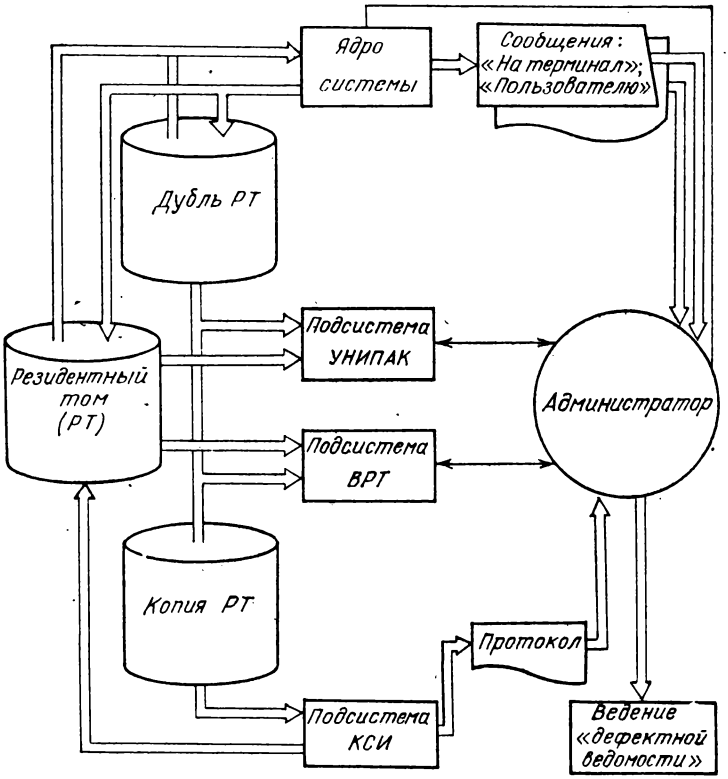


Рис. 5.19. Схема административного контроля и восстановления системы

Отсутствие старения и физического разрушения программ и, вследствие этого, расхода надежности при их эксплуатации, что характерно для аппаратуры [77], делает возможным повышение надежности и на этом этапе за счет усовершенствования средств

диагностики, улучшения реакций системы на внешние возмущения и исправления ошибок. В этом значительная роль отводится администратору системы. Его функции в обеспечении нормальной работы системы заключаются в следующем:

— регистрации и анализе ошибок, сбоев и отказов системы и устранении вызывающих их причин;

— восстановлении служебной информации в случае ее искажения.

На рис. 5.19 приведена схема, отражающая место и функции администратора в системе. Согласно схеме администратор

— осуществляет контроль за функционированием системы на основании сообщений, выдаваемых на операторский терминал и пользователю, а также протокола подсистемы контроля служебной информации (КСИ);

— устраняет искажение объекта РТ, если оно возникло, с помощью подсистемы УНИПАК и подсистемы восстановления резидентного тома (ВРТ), используя в качестве источников восстановления сам резидентный том, его дубль (ДРТ) или тома — копии резидентного тома;

— при таких отказах в ядре, как останов, зацикливание, зависание и т. п., определяет исполняемый в данный момент процесс и работающий модуль, место и причину отказа в модуле и исправляет ошибку, если она установлена.

Ведением «дефектной ведомости» осуществляются сбор и обобщение данных об отказах ядра и искажениях служебной информации для выявления случаев, требующих улучшения реакции системы на внешние возмущения (если они явились причинами отказов или искажений) либо доработки системы.

5.5. Схема поведения системы

Для обобщения методов обеспечения надежности УПД ДИАПАК рассматривается схема поведения системы, отражающая устойчивость работы системы в условиях внешних и внутренних возмущений.

Общие рамки правильного функционирования системы определяет граница области допустимых отклонений. Эта граница устанавливается заранее (при проектировании системы) на основании анализа других систем управления данными, опыта и интуиции разработчиков. Устойчивость системы обеспечивается, если протекающие в ней процессы не выходят за установленные границы. Модель поведения системы строится следующим образом. В пределах одной ЭВМ функционирование системы представляется как последовательное исполнение (согласно приоритетам) процессов. Пусть имеется n процессов, индекс процесса — i $(i = 1, \ldots, n)$.

Предполагается, что отказ при исполнении i-го процесса приводит к отказу системы, каждый процесс не влияет на другие процессы, изменение его состояния обусловлено внешними и внутренними возмущениями. Возможные состояния i-го процесса образуют вектор состояний, совокупность векторов — пространство состояний системы.

С учетом реакции системы на указанные возмущения, а также процедур контроля и восстановления пространство состояний можно разделить на подпространства:

— рабочего состояния процесса (i^1);

— состояния контроля и диагностики (i_1^2 при внешних возмущениях, i_2^2 при внутренних);

— состояния восстановления (i^3);

— состояния отказа (i^4).

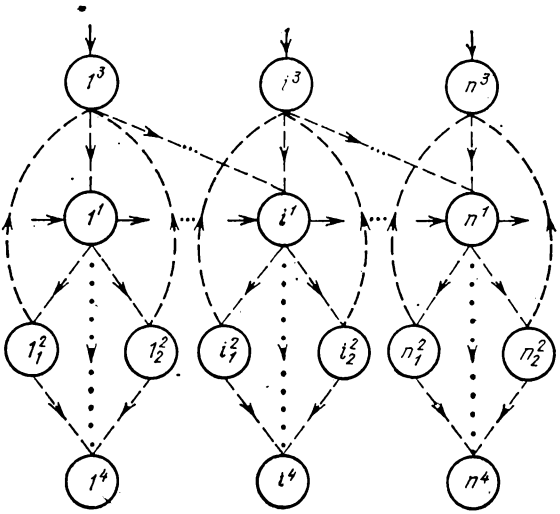


Рис. 5.20. Граф переходов между возможными состояниями системы

Соответствующая этому схема поведения системы представлена на рис. 5.20, где узлами обозначены возможные состояния процессов, дугами — переходы между состояниями. На этом рисунке сплошные стрелки означают переходы между процессами в рабочих состояниях, особо выделены непосредственные переходы из рабочих состояний в состояния отказов, пунктирные стрелки означают все остальные переходы, в том числе из состояний восстановления одного процесса к другому процессу. Переходы из i^1 в

i^2 или i^4 происходят случайно, во всех остальных направлениях — детерминированно. При этом переход из i^1 в i^4 обусловлен ошибками в ядре, приводящими к зацикливанию, останову, зависанию и т. п. (при коррекции с пульта ЭВМ возможен обратный переход, который не показан на рисунке), переход из i^2 в i^4 свидетельствует о возникновении такого возмущения, последствия которого недостаточно изучены.

Пребывание системы во всех состояниях, кроме i^1, сопряжено с непроизводительными затратами. Их анализ на основе рассмотренной модели приводится в следующей главе.

ГЛАВА 6

НЕКОТОРЫЕ ВОПРОСЫ ЭКСПЛУАТАЦИИ СИСТЕМЫ УПД ДИАПАК

6.1. Общие характеристики вычислительной обстановки

Широкое использование системы УПД ДИАПАК началось с 1978 г. Основные классы задач, при решении которых применялась эта система,— разнообразные задачи математической физики. Программы решения таких задач относятся к большим программам, которые характеризуются большим размером тела программы (десятки тысяч операторов фортрана или алгола); большими объемами перерабатываемой информации; большим временем исполнения программы; большой трудоемкостью при изготовлении (несколько десятков человеко-лет); большим временем жизни (как правило, программа эксплуатируется несколько — порядка 10 — лет) и, вследствие этого, многоразовыми модификациями программы (сотни и тысячи раз); большой трудностью в изучении и познании программы (почти всегда, на конкретном уровне всю программу не знает ни один из разработчиков) [5].

Вычислительная система, с помощью которой решались указанные выше задачи, представляет собой комплекс ЭВМ БЭСМ-6, имеющий общую память на магнитных дисках и терминальную сеть, включающую алфавитно-цифровые дисплеи. Развитие этой системы, вызванное ростом вычислительных потребностей задач и совершенствованием технологии вычислительного эксперимента, было связано главным образом с увеличением объема дисковой памяти и количества терминалов. За период с 1978 по 1982 гг. объем дисковой памяти и количество терминалов увеличились в несколько раз.

Операционная система ДИАПАК, под управлением которой работают ЭВМ комплекса, обеспечивает функционирование вычислительной системы в диалого-пакетном режиме и передачу данных

от одной ЭВМ к другой как через общую дисковую память, так и по каналу связи между оперативными памятями машин [9, 11].

Информационная база вычислительного комплекса, сформировавшаяся и находившаяся под контролем системы УПД ДИАПАК за период с 1978 по 1982 гг., характеризуется следующими тенденциями:

— постоянным ростом количества множеств и файлов;

— ростом общего объема информационного потока;

— регулярным обновлением более половины объема данных в течение сравнительно короткого времени;

— увеличением интенсивности поступления новых данных и, в связи с этим, увеличением количества уничтожаемых файлов (по истечении срока хранения или указанию из прикладной программы).

Ниже будет рассмотрен ряд вопросов, связанных с работой системы УПД ДИАПАК в период ее становления. К числу этих вопросов относятся описание практической методики управления информационными потоками и анализ методик обеспечения надежности системы и характеристик надежности системы. Будут приведены также расходы, возникающие при работе программных компонент системы УПД ДИАПАК, общий объем которых приближается к сотне тысяч команд БЭСМ-6.

Управление информационными потоками будет основываться на исследовании информационной базы и определении характеристик конкретных потоков. Оно состоит в объединении потоков, имеющих близкие характеристики, в более крупные групповые потоки и предоставлении им двухуровневой внешней памяти, сборе и анализе статистической информации о качестве обслуживания информационных потоков, перегруппировке потоков и изменении организации внешней памяти, если качество обслуживания снизилось.

Для оценки надежности системы УПД ДИАПАК в целом будут использоваться такие показатели, как среднее время безотказной (или бессбойной) работы и коэффициент готовности системы, о которых говорилось в гл. 2. Заметим сразу, что за анализируемое время эксплуатации системы эти показатели непрерывно менялись. Так, обнаружение и устранение программных ошибок внутри системы, повышение надежности работы оборудования, в частности устройств внешней памяти, повышение квалификации обслуживающего персонала (особенно администратора) и пользователей безусловно способствовали повышению надежности работы системы УПД ДИАПАК и вычислительной системы в целом. Развитие системы УПД ДИАПАК и, как следствие этого, появление новых программных ошибок, подключение новых устройств внешней памяти, относительная деквалификация при этом персонала и поль-

зователей приводили к относительному снижению уровня надежности работы системы УПД ДИАПАК и всей вычислительной системы.

6.2. Управление информационными потоками

Как было сказано выше, управление информационными потоками основывается на исследовании информационной базы и получении характеристик конкретных потоков. Для этого в метках файлов регистрируются сведения о некоторых событиях, например такие:

— дата создания файла (D_c);
— дата последнего обращения к файлу (D_0);
— количество обращений к файлу;
— дата *вталкивания* файла;
— количество *вталкиваний* файла;
— число обращений к файлу за время хранения его в пуле и др.

Суточная статистика, получаемая путем сбора сведений из меток файлов, накапливается и представляет собой интегрированные сведения по отдельным информационным потокам и районам, по групповым потокам и пулам, по всей информационной базе. Дату сбора статистики будем называть *текущей датой* и обозначать $D_т$. Анализ этой статистики позволяет исследовать количественный состав информационной базы и ее характеристики, изучить свойства отдельных информационных потоков, проследить динамику поведения базы в целом в течение определенного периода времени, оценить эффективность использования внешней памяти и прогнозировать потребность в ней.

Рассмотрим методику управления информационными потоками на конкретном примере — фрагменте, взятом из практики.

Пусть имеются три множества A, B и C, причем A и B — множества результатных величин, рассчитываемых соответственно по двум разным программам, а C — множество программ серийного решения задач математической физики. Общий объем хранимых данных в множестве A составляет 10^8 байт, в множестве B — $4{,}8 \cdot 10^7$ байт, в множестве C — $3{,}2 \cdot 10^7$ байт. Функция распределения данных в этих множествах по времени их хранения $T_{хр} = D_т - D_0$ (рис. 6.1) показывает, что соответствующие им информационные потоки заметно отличаются друг от друга. Скорости информационных потоков равны: в множестве A — $3 \cdot 10^6$ байт/сут., в множестве B — $1{,}2 \cdot 10^5$ байт/сут., в множестве C — 10^4 байт/сут. Данные находятся на хранении в течение разного периода времени: в множестве A — 1,5 месяца, в множестве B — 1,5 года, в множестве C — несколько лет. Следовательно, данные этих множеств

должны быть размещены в разных отдельных областях памяти нижнего уровня.

Для определения целесообразности использования двухуровневой системы памяти для этих множеств исследуем функцию распределения данных по времени их использования $T_{исп} = D_0 - D_0$ (рис. 6.2). Из приведенных графиков видно, что в множестве A

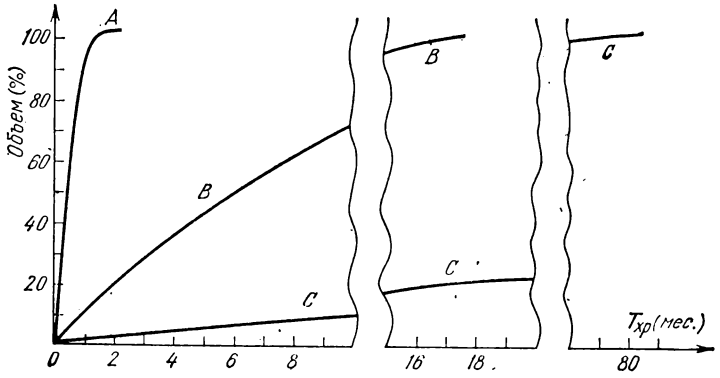


Рис. 6.1. Распределение данных по времени хранения

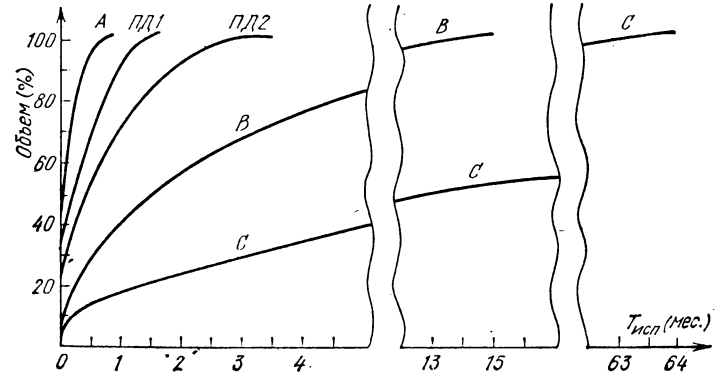


Рис. 6.2. Распределение данных по времени использования

использование данных в основном завершается к концу третьей недели с момента поступления их на хранение, т. е. что при использовании двухуровневой памяти потребуется задержка файлов этого множества в памяти верхнего уровня на этот срок. Это в свою очередь может быть обеспечено объемом памяти $\sim 7 \cdot 10^7$ байт. Последняя величина получена на основании функции распределения данных множества A по времени хранения (см. рис. 6.1).

Подобного вывода нельзя сделать для множеств B и C, так как активный период для этих множеств велик и почти соизмерим со временем хранения данных. В частности, в множестве B он достигает одного года, а в множестве C — нескольких лет. В этих случаях для исследования привлекается функция $a(\tau, t)$, рассмотренная в гл. 1.

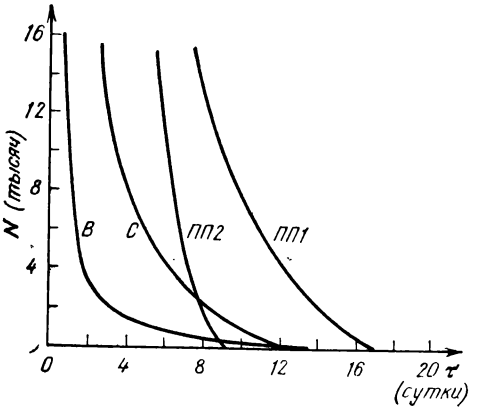


Рис. 6.3. Зависимость $N(\tau)$

Исследование зависимости количества скачков N, которые имеет функция $a(\tau, t)$ для множеств B и C, от параметра τ (рис. 6.3) приводит к выводу, что для удовлетворительного обслуживания информационных потоков необходимо обеспечить задержку файлов в памяти верхнего уровня для множества B в течение 15 сут. от даты последнего обращения к ним, а для множества C — в течение 12 сут. Привлекая для дальнейшего исследования функцию распределения данных по времени «простоя» $T_п = D_т - D_о$ (рис. 6.4), находим объем памяти, необходимый для этих данных на верхнем уровне: для множества B он равен 10^7 байт, для множества C — $0{,}9 \cdot 10^7$ байт. Отсюда следует целесообразность применения двухуровневой системы памяти и для множеств B и C. Тем не менее, если к файлу не будет обращения в течение длительного периода времени, то он автоматически вытесняется с верхнего уровня памяти на нижний, а при возобновлении использования этот файл будет снова передан на верхний уровень памяти.

При решении вопроса об объединении информационного потока с одним из уже существующих групповых потоков в первую очередь учитывается алгоритм замещения файлов, обслуживающий данный поток. При выборе подходящего группового потока среди тех, которые обслуживаются нужным алгоритмом, для каждого такого потока строится зависимость, описанная выше, и на основе

сравнения этих зависимостей выбирается наиболее подходящий групповой поток. В частности, на рис. 6.2 приведена функция распределения данных по времени использования для групповых потоков ПД1 и ПД2, а на рис. 6.3 приведена зависимость $N = N(\tau)$ для групповых потоков ПП1 и ПП2.

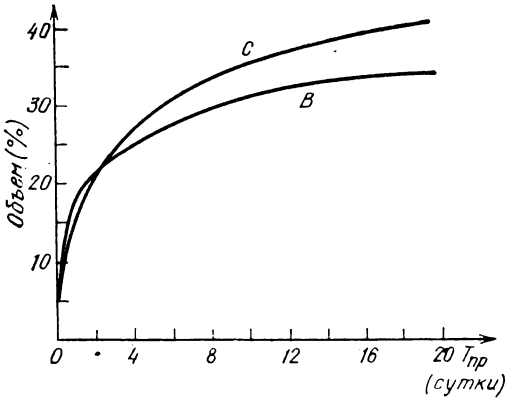


Рис. 6.4. Распределение данных по времени «простоя»

Легко видеть, что наиболее подходящим для информационного потока A является групповой поток ПД1, в котором более 80% данных используется в течение первых трех недель с момента создания файла, а не поток ПД2. Информационные потоки B и C лучше объединить с групповым потоком ПП1, для которого задержка файла в пуле от момента последнего использования требуется в течение 17 сут., а не с потоком ПП2, для которого это значение равно 9 сут.

Качество обслуживания информационных потоков оценивается по фактически предоставляемому объему памяти на верхнем уровне и по числу передач файлов с нижнего уровня памяти на верхний уровень памяти, т. е. по числу вталкиваний файлов: чем меньше вталкиваний, тем лучше обслуживается поток.

Например, для группового потока ПД1 требуется пул емкостью $1{,}6 \cdot 10^7$ байт. Фактически этому потоку была выделена память емкостью $1{,}6 \cdot 10^8$ байт, что обеспечило задержку файлов в пуле на 22 сут. от даты создания файла. Число вталкиваний файлов при этом редко превышало 15 в сутки. Другой пример, для группового потока ПП1 требуется $1{,}15 \cdot 10^8$ байт памяти на верхнем уровне. Фактически этому потоку была выделена память емкостью $1{,}3 \cdot 10^8$ байт, что обеспечило задержку файлов в пуле на 30 сут. от даты последнего обращения к ним. Число вталкиваний файлов редко превышало 10 в сутки. Учитывая, что число вталкиваемых

файлов в групповых потоках ПД1 и ПП1 составляет в среднем 2% от общего числа файлов, используемых в этих потоках в течение суток (300 и 200 соответственно), можно утверждать, что обслуживание этих потоков вполне удовлетворительное.

С информационным потоком A был проведен эксперимент: на верхнем уровне ему была выделена память различного объема. При объеме памяти $7{,}2 \cdot 10^7$ байт происходило одно вталкивание файла в течение 5 сут. Уменьшение объема памяти до $6{,}9 \cdot 10^7$ байт незначительно увеличило число вталкиваний файлов до одного в сутки. Дальнейшее уменьшение объема памяти до $6{,}0 \cdot 10^7$ байт, что меньше расчетной нормы на 10^7 байт, привело к резкому увеличению числа вталкиваний файлов — до 24 в сутки. Результаты этого эксперимента иллюстрируют механизмы методики управления информационными потоками и оценки объема памяти, необходимой этим потокам на верхнем уровне.

Автоматизация функций управления данными влечет определенные затраты ресурсов вычислительной системы. К ним относятся:

— время центрального процессора, используемое компонентами системы;

— загрузка каналов обмена с внешней памятью при работе со служебной информацией;

— простои центрального процессора из-за ожидания выполнения заявок системой управления данными;

— часть внешней памяти для хранения служебной информации.

Измерения показали, что эти затраты весьма незначительны. Так, затраты времени центрального процессора по отношению к среднему времени функционирования системы составляют не более 5%, число обменов с магнитными дисками, выполняемых модулями УПД ДИАПАК, составляет 23% от всех обменов с МД, нагрузка на каналы обмена с магнитными барабанами составляет не более 2%, время простоя центрального процессора из-за ожидания выполнения заявок составляет 1—1,5%. Затраты дисковой памяти на хранение служебной информации вычисляются по формуле

$$V = 1{,}72 \cdot 10^6 + 480 \cdot n \text{ (байт)},$$

где n — число файлов, находящихся на хранении. В нашем случае эти затраты составляют 0,02% от общего объема памяти для хранения данных.

6.3. Оценка методов обеспечения надежности

Рассмотрим оценку эффективности используемых в УПД ДИАПАК методов обеспечения надежности и надежности этой системы в целом.

При анализе методов используются показатели, позволяющие оценить эти методы с разных точек зрения, и приводятся затраты, связанные с их реализацией.

Для обеспечения надежности в УПД ДИАПАК используются методы:

— контроля входных данных;

— обнаружения внешних и внутренних возмущений;

— обеспечения сохранности служебной информации и файлов;

— восстановления служебной информации, файлов и функционирования системы.

Начнем с анализа метода контроля входных данных. Этот метод широко используется в программных системах. Он заключается в обнаружении ошибок пользователя, оператора и администратора при взаимодействии с системой, регистрации ошибок, их диагностики и выдаче соответствующих сообщений. Метод характеризуется:

— временем обнаружения ошибки $T_{\text{о}}^{1}$;

— вероятностью обнаружения ошибки $P_{\text{о}}^{1}$;

— вероятностью ложного обнаружения ошибки $P_{\text{л}}^{1}$;

— точностью диагностики;

— затратами на проведение контроля.

$T_{\text{о}}^{1}$ — это отрезок времени между моментом приема операции (здесь и ниже понимаются операции управления районом или файлом) и моментом обнаружения ошибки при выполнении этой операции; $T_{\text{о}}^{1}$ является характеристикой потерь, связанных с той работой, которая затрачена системой на выполнение ошибочной операции. Таким образом, чем раньше будет обнаружена ошибка, тем меньше величина потерь. Расчет $T_{\text{о}}^{1}$ осуществляется отдельно для синтаксических, логических и других — *фатальных* ошибок — на основании статистических данных о числе таких ошибок, общем числе операций, выполненных системой, и средней *цене* (среднем времени выполнения) операции. Как известно, в цену операции существенный вклад вносят обмены с РТ и магнитным барабаном (МБ) ОС, где располагаются модули системы УПД ДИАПАК.

Зная общее количество обменов с РТ ($К^{РТ}$), МБ ОС ($К^{МБ}$) и число операций ($Ч^{ОП}$), выполненных в течение исследуемого периода работы системы, можно определить среднее время выполнения операции

$$T = \frac{К^{РТ} \times t_{РТ} + К^{МБ} \times t_{МБ} + t_{ЦП}}{Ч^{ОП}},$$

где $t_{РТ}$ — среднее время выполнения обмена с МД, $t_{МБ}$ — среднее время выполнения обмена с МБ, $t_{ЦП}$ — процессорное время работы

системы (оно по сравнению с двумя первыми членами числителя существенно мало и в последующих оценках не участвует). Эта формула дает среднее время выполнения операции $T \sim 500 \cdot 10^{-3}$ с. Заменой величин $Ч^{оп}$, $К^{РТ}$, $К^{МБ}$ величинами $Ч^{со}$ — числом операций с синтаксическими ошибками, $К^{РТ}_{со}$ и $К^{МБ}_{со}$ — количеством обменов соответственно с РТ и МБ при выполнении операций с такими ошибками и аналогичными величинами для логических и фатальных ошибок было получено:

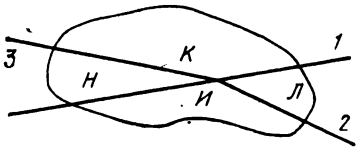


Рис. 6.5. Распределение операций по ошибкам

— среднее время, необходимое для обнаружения синтаксической ошибки $\sim 25 \cdot 10^{-3}$ с;

— среднее время обнаружения логической ошибки $\sim 290 \cdot 10^{-3}$ с;

— среднее время обнаружения фатальной ошибки $\sim 400 \cdot 10^{-3}$ с.

Зная количество всех ошибок и их распределение по типам, можно определить $T^1_о$. В нашем случае оно равно $220 \cdot 10^{-3}$ с.

Вероятности $P^1_о$ и $P^1_л$ определяют точность метода. Чтобы оценить их, необходимо все выполненные за исследуемый период операции разделить на четыре части (рис. 6.5). Сначала определяются операции, в которых обнаружена ошибка. Такие операции составляют первую часть, остальные операции — вторую. В первой части выделяются операции с истинными и ложными ошибками (И, Л), во второй — операции с необнаруженными ошибками (Н) и корректные, действительно безошибочные операции (К). В соответствии с указанным разделением можно определить (рассматривая вероятность как идеализацию статистической частоты случайного события), что

$$P^1_о \approx \frac{И}{И + Н}, \qquad P^1_л \approx \frac{И}{Л + И}.$$

Отмеченное на рисунке разделение по линии *1* выполняется системой, а по линии *2* осуществляется администратором при разборе аварийного завершения выполнения операции по синтаксической или логической ошибке (фиксируются случаи ложного обнаружения), по линии *3* выполняется администратором при анализе сбоев или отказов системы, искажений служебной информации и файлов (фиксируются случаи пропуска ошибочной информации). Значения вероятностей обнаружения синтаксических, логических и фатальных ошибок, вычисленные по этим формулам, близки к 1, а вероятностей ложного обнаружения — к 0.

Точность диагностики характеризуется степенью локализации ошибки. Критерий оценки прост: если диагностическое сообщение

содержит информацию, достаточную для исправления ошибки, то точность оценивается единицей; если место ошибки определено неверно или необходима дополнительная информация для его определения, точность оценивается нулем. Для синтаксических и логических ошибок оценка точности осуществляется администратором на основе протокола, полученного после прогонки специального теста — имитатора ошибок и протоколов пользователя (ошибочная операция и текст сообщения об ошибке выдается пользователю на АЦПУ). В настоящее время достигнутая точность практически равна единице. Сообщение о фатальной ошибке в основном предназначено администратору (выдается на операторский терминал) и указывает на характер ошибки и объект системы, сбой или отказ которого привел к возникновению такой ошибки.

Процедура контроля входных данных совмещена с вычислительными операциями, поэтому затраты на ее проведение незначительны: в командах они составляют всего 1,4% от общего объема системы, в обменах с внешней памятью — менее 1% от общего количества обменов, выполняемых системой за сутки.

Рассмотрим теперь методы обнаружения внешних и внутренних возмущений. Эти методы разделяются на *оперативные* и *автономные*. Они направлены на обнаружение, регистрацию и диагностику некорректных заказов от модулей ОС, искажений служебной информации в ОП и на РТ, структуры хранения последовательных, индексно-последовательных и библиотечных файлов, а также собственных программных ошибок. Методами оперативного обнаружения обследуется состояние служебной информации и структуры хранения файлов. Автономные методы являются дополнительными для контроля служебной информации и структуры хранения файлов, поскольку эти компоненты во многом определяют нормальное функционирование системы.

«Перегрузки» системы (исчерпание ресурсов РТ, переполнение таблицы открытых файлов и т. п.) могут быть причиной ее сбоев. Их обнаружение также возлагается на рассматриваемые методы.

Методы обнаружения характеризуются следующими показателями:

— временем обнаружения возмущения T_0^2;
— вероятностью обнаружения возмущения P_0^2;
— вероятностью ложного обнаружения возмущения;
— точностью диагностики;
— затратами на обнаружение возмущения;
— оперативностью индикации состояния системы;
— правильностью реакции на возмущение.

Смысл первых пяти показателей тот же, что и для аналогичных показателей метода контроля входных данных. Особенность

их состоит в трудности измерения и большом разбросе значений. Достаточно сложно, а в ряде случаев невозможно организовать автоматическую регистрацию и сбор соответствующих статистических данных как из-за разнообразия контролируемых возмущений, так и из-за фатальности операционной ситуации, например, при остановах, зацикливаниях или зависаниях системы (не выполняется запланированная работа и прерывается связь с оператором), отказах РТ и тома дубля РТ и др. Поэтому для определений значения первых трех показателей используются информация из «дефектной ведомости», статистические данные о работе внешних запоминающих устройств, о перевызовах ОС после отказа, сведения, выдаваемые на операторский терминал, протокол подсистемы контроля служебной информации, сведения из журнала работы ВС, заполняемого оператором. При этом получены следующие результаты.

При оперативном обнаружении T_0^2 невелико (составляет в среднем $200 \cdot 10^{-3}$ с, а при сбоях в процессе обмена с внешней памятью, при перегрузках системы и искажениях таблицы открытых файлов — $60 \cdot 10^{-3}$ с). Однако этот метод не позволяет обнаружить все искажения служебной информации на РТ или все искажения структуры хранения файлов. Применение автономного контроля решает эту задачу, но при этом T_0^2 определяется интервалом времени между последовательными процедурами копирования РТ и файлов. На практике значение T_0^2 для служебной информации варьируется от 12 до 24 часов (администратор при необходимости может выбрать любой интервал копирования), для файлов T_0^2 устанавливается пользователем.

Вероятность P_0^2 постепенно приближается к 1, исключение составляют случаи программных ошибок, приводящих к останову или зацикливанию. Однако они очень редки (менее 10 за год). Случаи ложного обнаружения не наблюдались.

Требования к точности диагностики обнаруженных возмущений более высоки, чем для метода контроля входных данных. Это объясняется сложностью локализации места и уточнения причины неисправности и большими потерями при запаздывании принятия соответствующих мер особенно в условиях многомашинного комплекса. Кроме того, в ряде случаев восстановление невозможно или нецелесообразно, пока не будут определены последствия возмущения или не будет установлена причина неисправности. Поэтому можно дать только качественную оценку: в целом, точность диагностики обнаруженных возмущений удовлетворительна. Дополнительный анализ необходим только при разборе таких ситуаций, как останов, зацикливание и зависание системы (это занимает в среднем 20 мин работы опытного администратора системы

за пультом ЭВМ). В остальных случаях в диагностическом сообщении указывается «имя» процесса, характер возмущения, место обнаружения и координаты объекта служебной информации, если произошло его искажение.

Затраты на обнаружение возмущений и их диагностику составляют: в командах — 18% от общего объема системы, в обменах с внешней памятью — 9% от общего числа обменов, выполняемых системой за сутки.

Индикация состояния системы является одним из важных условий оперативного управления и слежения за ее работой, осуществляемых оператором и администратором. После обнаружения возмущения диагностическое сообщение выдается на операторский терминал, а отклонения в работе системы фиксируются на специальном пульте индикации — световом табло. Появление такого сигнала на табло рассматривается как поступление заявки на обслуживание.

Реакция на возмущение определяется после тщательной оценки его последствий для функционирования системы. При проектировании надежности возможна и осуществима только самая грубая оценка последствий. Поэтому первоначально в конструкцию системы закладываются примитивные блоки реакции на возмущения: программный останов, аварийное завершение задачи. В процессе эксплуатации оценка возможных последствий прогнозируемого возмущения уточняется; совершенствуется реакция — она становится более гибкой, т. е. способствует уменьшению или предотвращению потерь при выходе из аварийных ситуаций.

Рассмотрим методы обеспечения сохранности служебной информации и файлов. Они состоят главным образом в дублировании и копировании служебной информации, копировании и ведении поколений файлов и характеризуются:

— вероятностью переноса искажений основного экземпляра в дубль или копию;

— вероятностью успешного завершения копирования служебной информации или файла;

— затратами на проведение процедур дублирования, копирования и ведения поколений.

Для реализации этих методов необходимы ресурсы внешней памяти:

— том для дубля РТ (возможно совмещение основной и дубль-области на самом РТ, однако это повышает вероятность искажения информации);

— тома для копий;

— отдельные ленточные районы [62] для организации копирования наиболее важных файлов коллективного использования.

Дублю РТ присваивается статус системного тома, чтобы обеспечить защиту его от задач пользователя. Том копии требуется только на момент копирования и автономного контроля служебной информации. Тома для дубля РТ и его копий в последующем будут называться *служебными томами*.

Дублирование служебной информации осуществляется автоматически. Процесс копирования служебной информации инициируется директивой оператора, выполняемой в любой момент времени. Копирование файла осуществляется средствами подсистемы УНИПАК. Этот процесс инициируется директивой пользователя во время сеанса связи с этой подсистемой, в ходе диалога пользователь уточняет задание на копирование. По завершении всего процесса пользователю выдается сообщение о результате копирования. Для ведения поколений файла достаточно при его определении указать число поколений. Формирование поколений и поддержание их числа в указанных пределах осуществляется автоматически.

Вероятность переноса искаженной служебной информации при дублировании и копировании может достигать величины 0,5 (т. е. допускается самый худший случай и учитываются возможности оперативного обнаружения искажений). На практике эта вероятность близка к нулю. Следует добавить, что процесс копирования считается завершенным, если не обнаружено искажений служебной информации при ее контроле подсистемой КСИ.

При копировании файлов в условиях двухуровневой памяти исключается перенос из пула в район искаженных по «вине» системы последовательных, индексно-последовательных и библиотечных файлов. Это достигается проверкой целостности структуры хранения таких файлов, осуществляемой в процессе копирования.

Прерывание процесса копирования служебной информации или файлов (например, из-за сбоя или отказа ЭВМ) не имеет отрицательных эффектов. Поэтому вероятность успешного завершения копирования близка к 1 (исключение составляют случаи одновременного отказа РТ и ДРТ при копировании служебной информации и отказа тома пула с файлом при копировании файла).

Затраты на обеспечение сохранности служебной информации и файлов составляют: в командах — 6,3% от общего объема системы, в обменах с внешней памятью — 24% от общего числа обменов, выполняемых системой.

Перейдем к рассмотрению методов восстановления служебной информации, файлов и функционирования системы после возмущений.

Результаты обнаружения и диагностики возмущений передаются блокам реакции, которые регистрируют возмущения и определяют дальнейшее поведение системы. При этом в одних случаях решение принимает сама система и функционирование ее продол-

жается, так как при возникновении прогнозируемых сбоев и искажений проводится автоматическое восстановление служебной информации. В других случаях принимает решение и выбирает функцию восстановления лицо, которому адресовано сообщение. Для

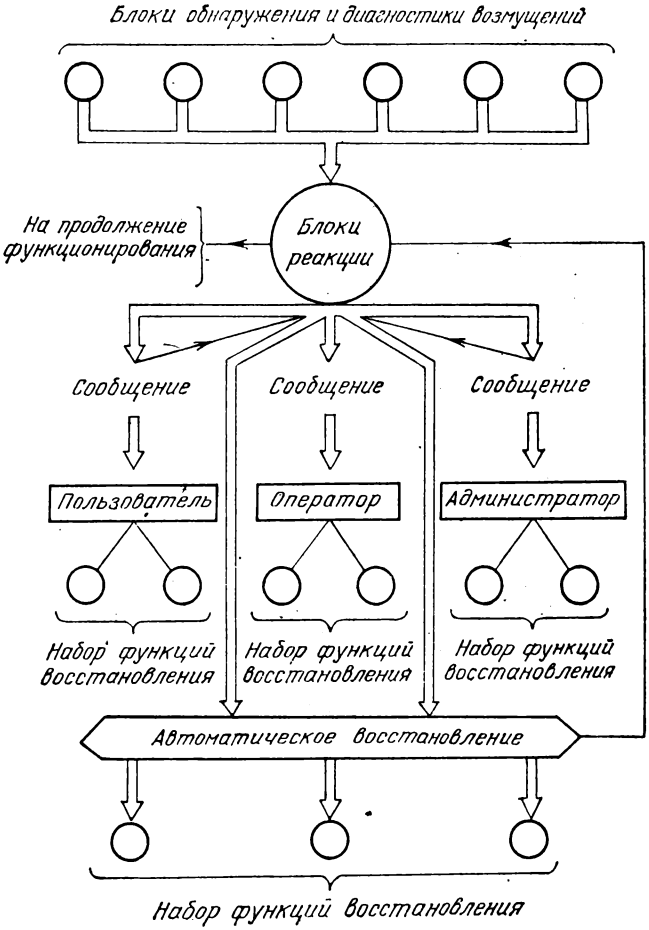


Рис. 6.6. Схема подготовки решения и определения операций восстановления

каждого из указанных лиц имеется свой набор функций, реализуемых прикладными подсистемами. Схема подготовки и проведения операции восстановления представлена на рис. 6.6.

Методы восстановления служебной информации, файлов и функционирования системы характеризуются:

— вероятностью полного восстановления $P_{в}^{4}$;

— потерями файлов при восстановлении служебной информации;

— длительностью восстановления $T_{в}^{4}$;

— затратами на восстановление.

При восстановлении служебной информации значение $P_{в}^{4}$ зависит от характера искажения конкретного объекта и источника восстановления. Если сохранена информация на ДРТ, гарантируется полное восстановление. Если искажен каталог файлов, а в качестве источника восстановления может быть использован только КРТ, то неизбежны потери тех файлов, которые были созданы или увеличились в размере в результате модификации после копирования РТ. В табл. 6.1 приводятся потери, связанные с восстановлением служебной информации.

При восстановлении файла значение $P_{в}^{4}$ зависит от того, насколько «свежи» копия, последнее поколение или экземпляр, хранящийся в районе (если восстанавливаемый файл находится в пуле).

При восстановлении функционирования системы после отказа (останов, зацикливание или зависание) значение $P_{в}^{4}$ зависит от квалификации администратора и степени искажения вычислительного процесса. Потери при восстановлении определяются количеством задач, снятых из-за отказа или сбоя системы.

При автоматическом восстановлении служебной информации значение $T_{в}^{4}$ достаточно мало (оценивается временем исполнения нескольких обменов с МД $60 \cdot 10^{-3}$—$240 \cdot 10^{-3}$ с). В случае восстановления, осуществляемого администратором, значение $T_{в}^{4}$ зависит от тех же факторов, что и $T_{в}^{4}$, а также от режима восстановления: с сохранением или блокировкой функционирования системы (режимы 1, 2). В режиме 1 длительность восстановления не влияет на интегральные показатели надежности системы (в частности, на коэффициент готовности $K_{г}$). В режиме 2 (после катастрофических отказов) все ЭВМ комплекса «останавливаются», на одной из них перевызывается автономная ОС (с блокировкой включения задач в решение) и осуществляется восстановление, длительность которого может изменяться от нескольких минут до нескольких часов.

Длительность восстановления файла определяется суммой двух величин: T' — времени переписи копии в основной файл (или вталкивания файла из района в пул, если искажен экземпляр в пуле, и имеется версия файла в районе, либо времени замены поколения) и T'' — времени получения состояния файла, предшествующего искажению. При замене поколения $T' \sim 420 \cdot 10^{-3}$ с.

В случае переписи копии (или вталкивания) оценка T' осуществляется по формуле

$$T' = N \times \left(T^{к}_{чт} + T^{о}_{зп}\right),$$

где N — размер файла в зонах, $T^{к}_{чт}$ — время чтения зоны копии файла, $T^{о}_{зп}$ — время записи зоны основного файла. Если выполнения одной из этих процедур достаточно для восстановления файла до состояния, предшествующего искажению, то $T'' = 0$.

Таблица 6.1

Возможные потери файлов при восстановлении объектов служебной информации на РТ

Восстанавливаемый объект	Источник восстановления	Потери файлов
Метка РТ	ДРТ, КРТ	нет
Таблица распределения ресурсов РТ	ДРТ Протоколы КСИ, сверки ресурсов РТ	нет 0,3% от количества созданных за сутки файлов
Каталог файлов	ДРТ КРТ	нет 0,3—0,5% от количества созданных или модифицированных за сутки файлов
Каталог районов и пулов	ДРТ КРТ	нет нет
Таблица распределения архивных томов	ДРТ КРТ	нет нет
Таблица свободных зон района (пула)	ДРТ Протоколы КСИ, сверки пространства в районе или пуле	нет 0,3—0,5% от количества созданных или модифицированных за сутки файлов

В противном случае необходим перерасчет по программе пользователя и T'' может составить несколько минут (или быть одного порядка с T'), а в исключительных случаях — несколько часов, и тогда оно становится определяющим в оценке длительности восстановления файла.

Длительность восстановления функционирования системы после отказа (останов, зацикливание, зависание) определяется

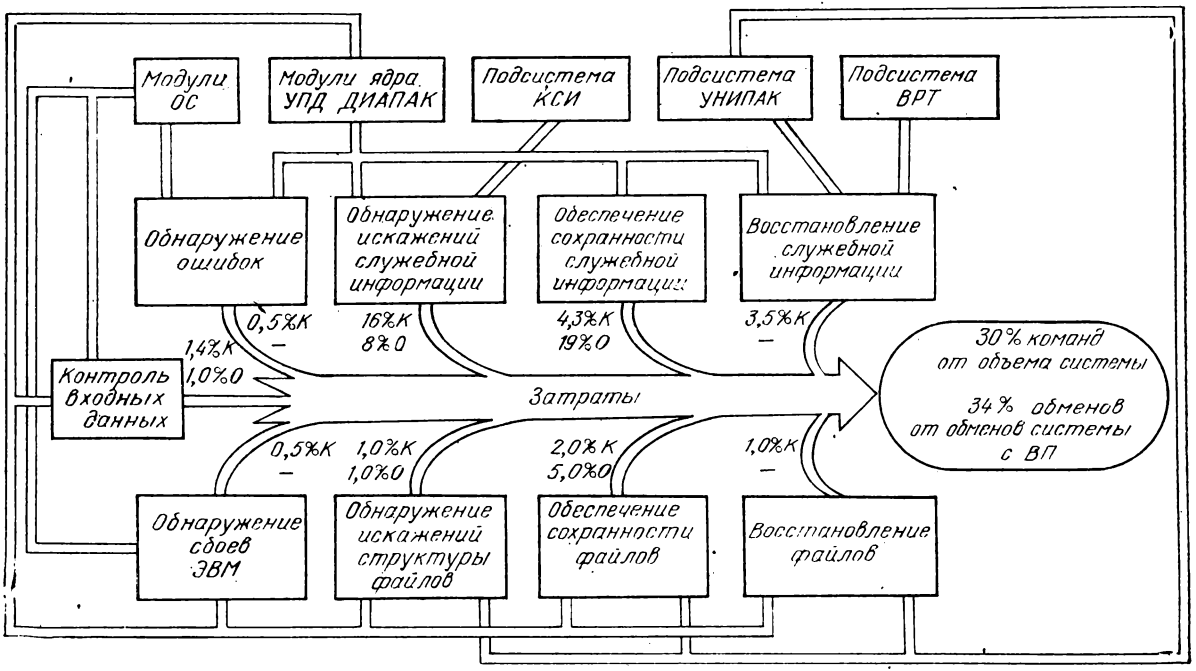


Рис. 6.7. Распределение затрат на обеспечение падежности системы УПД ДИАПАК (сокращения 1,4% *K*, 1% *O* означают относительные затраты в командах, обменах)

временем локализации ошибки и регистрации всех необходимых данных для установления причины ее возникновения. Это занимает в среднем около 20 мин работы опытного администратора за пультом ЭВМ.

Затраты на реализацию методов восстановления служебной информации и файлов составляют в командах 4,5% от общего объема системы.

Распределение затрат, связанных с реализацией всех рассмотренных методов, приводится на рис. 6.7.

6.4. Характеристики надежности системы

С самого начала широкого использования системы УПД ДИАПАК выявился ряд недостатков. К их числу в первую очередь относились слабая защита системы от ошибок пользователя и неточность их диагностики, отрицательное влияние перегрузок на нормальное функционирование системы, «грубость» реакции системы на внешние возмущения, вследствие чего происходили потери времени счета задач, запаздывание обнаружения искажений служебной информации и файлов.

В 1978 г. устранялись отмеченные недостатки, осуществлялась регистрация сбоев и отказов в дефектной ведомости. За период с 1978 по 1979 гг. преобладающее число сбоев и отказов системы происходило из-за собственных программных ошибок — 83%, в то время как по вине ОС — 7%, обслуживающего персонала — 1%, аппаратуры — 9%. Среднее время безотказной работы составляло 75 ч для комплекса ЭВМ, коэффициент готовности — 0,87.

В 1979 г. продолжалось развитие системы: был реализован базисный метод доступа к данным, введены новые экстракоды для организации работы системы УПД ДИАПАК в диалоговом режиме, совершенствовались алгоритмы управления двухуровневой внешней памятью. Все это привело к снижению надежности системы: возросло число искажений служебной информации (вследствие чего участились случаи пересечения файлов), остановов и зацикливаний в ядре системы и зависаний задач пользователей (задачи оставались длительное время в состоянии ожидания конца выполнения операций; только внешним вмешательством удавалось активизировать их). Для этого периода статистика сбоев и отказов системы УПД ДИАПАК выглядит следующим образом: из-за программных ошибок — 96%, в то время как по «вине» ОС — 1%, обслуживающего персонала — 1%, аппаратуры — 2%. Среднее время безотказной работы составляло 53 ч, коэффициент готовности — 0,81. Возрастание числа ошибок объяснялось внедрением новых возможностей системы и усовершенствованием ряда ее внутренних алгоритмов.

В 1980 г. после исправления ошибок в системе ее надежность заметно улучшилась: среднее время безотказной работы возросло до 174 ч, коэффициент готовности составил 0,97. Статистика сбоев и отказов системы УПД ДИАПАК была такова: из-за программных ошибок — 66%, в то время как по «вине» ОС — 16%, обслуживающего персонала — 7%, аппаратуры — 11%. Возрастание числа сбоев и отказов по «вине» ОС и аппаратуры объясняется внедрением новых типов внешних запоминающих устройств (дисков большой емкости).

К середине 1981 г. среднее время безотказной работы достигло 230 ч, коэффициент готовности составил 1.

Результирующий график изменения среднего времени безотказной (а с 1981 г.— бессбойной) работы, а также коэффициента готовности приводится на рис. 6.8.

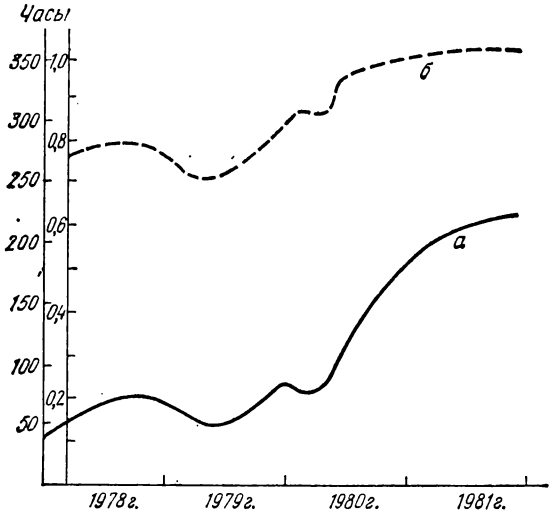


Рис. 6.8. Изменение среднего времени безотказной работы (кривая *а*) и коэффициента готовности (кривая *б*) системы УПД ДИАПАК

Последствия сбоев и отказов системы оцениваются количеством потерянных файлов и аварийно завершившихся задач. За 1978—1981 гг. потери файлов составили 0,04% от общего числа обработанных файлов, количество задач, снятых по «вине» УПД ДИАПАК, составило всего 0,15% от общего числа решенных задач (для сравнения: по «вине» аппаратуры — 1,42%, ОС ДИАПАК — 0,72%).

Результаты эксплуатации системы УПД ДИАПАК, описанные в данной главе, свидетельствуют:

— о достаточной гибкости управления потоками данных и расчета объемов памяти для данных, что обеспечивает эффективную работу системы;

— о работоспособности методов обеспечения надежности, приведших к высокому уровню надежности самой системы;

— о вполне приемлемых, мало снижающих эффективность системы расходах процессорного времени, памяти и нагрузках на каналы обменов.

ПРИЛОЖЕНИЕ

ЯЗЫК ВЗАИМОДЕЙСТВИЯ ПОЛЬЗОВАТЕЛЯ С СИСТЕМОЙ

В данном приложении приводится более подробное описание элементов языка системы УПД ДИАПАК.

Система предоставляет пользователю набор операций, с помощью которых можно:

— открыть район (ОТР);
— закрыть район (ЗКР);
— уничтожить район (УНР);
— определить файл (ОПФ);
— открыть файл (ОТФ);
— закрыть файл (ЗКФ);
— уничтожить файл (УНФ);
— создать раздел библиотечного файла (СРФ);
— найти раздел и открыть его (НРФ);
— закрыть раздел (ЗРФ);
— уничтожить раздел (УРФ);
— выбрать запись из файла;
— внести новую или модифицировать существующую запись в файле;
— уничтожить запись в файле;
— обновить файл.

В дальнейшем изложении операции управления районами, файлами и разделами называются *процедурами доступа.* Отметим, что операции над множествами совмещены с операциями над районами. Имя района одновременно становится именем множества.

§ 1. Процедуры управления районами, файлами, разделами

В системе реализованы два способа задания процедур доступа:

— при помощи разделов паспорта задачи (статическое задание);
— при помощи экстракодов (динамическое задание).

Каждая процедура имеет поле операции и поле операндов, разделенных хотя бы одним пробелом. В поле операции пробелы недопустимы, в поле операндов пробелы игнорируются. Признаком конца процедуры служит символ «надчеркивание». В поле операции указывается название процедуры.

Поле операндов содержит один или несколько операндов, отделенных друг от друга запятыми. Операнды могут быть позиционными и ключевыми, списковыми, обязательными и необязательными. Некоторые операнды имеют значения, принимаемые по умолчанию.

При описании процедур выбрана ключевая форма записи, при этом операнды следуют согласно их позициям. Те операнды, которые могут быть списковыми, помещаются в круглые скобки. Необязательные операнды заключены в квадратные скобки. В угловых скобках (<, >) дается смысловое значение операнда.

Открытие района:

ОТР РАЙ = <имя района>
КРН = <код района>⁻

<имя района> — идентификатор, содержащий от одного до шести символов;

<код района> — логический номер, с помощью которого процедуры управления файлами адресуются к данному району.

Закрытие района:

ЗКР КРН = <код района>⁻

По данной процедуре осуществляется закрытие района с указанным кодом.

Уничтожение района:

УНР РАЙ = <имя района>
[КЛУ = <ключ уничтожения>]⁻

<ключ уничтожения> — код, который сравнивается с эталоном, заданным при определении района.

Уничтожить район имеет право только его владелец. К моменту исполнения процедуры с районом не должна работать ни одна задача и он не должен содержать файлов.

Определение файла:

ОПФ КРН = <код района>
ФАЙ = <имя файла>
КФЛ = <код файла>
[ПОК = <число поколений>]
[КЛУ = <ключ уничтожения>]
([ПРВ = <пространство под файл>])
([ДИС = <статус и диспозиция файла>])
[СХР = <срок хранения файла>]
[ОРГ = <организация файла>]
[ФЗП = <формат записи>]⁻

⟨имя файла⟩ — последовательность от одного до четырех простых имен, отделенных друг от друга символом «точка». Простое имя может содержать от одного до шести символов. Первым именем должен быть идентификатор, остальные могут начинаться как с буквы, так и с цифры, но не могут быть нулевыми.

⟨код файла⟩ — логический код, с помощью которого устанавливается принадлежность обрабатываемых записей к данному файлу.

⟨число поколений⟩ — служит для указания необходимости ведения поколений файла.

⟨ключ уничтожения⟩ — эталон для одноименного операнда из процедуры уничтожения файла.

⟨пространство под файл⟩ — списковый операнд, содержащий от двух до четырех подоперандов. Общий вид операнда:

ПРВ = (P, L, M, N)

где P — размер основного экстента памяти (в записях); L — длина записи (в словах); M — размер дополнительного экстента (в записях); N — количество дополнительных экстентов (не более трех). Суммарный объем пространства не должен превышать четырех томов. Если информация о дополнительных экстентах отсутствует, то операнд имеет вид:

ПРВ = (P, L)

Если операнд опущен, то по умолчанию будут приняты: P = 1, L = 1024.

⟨статус и диспозиция⟩ — списковый операнд. Статус файла устанавливает указатель текущей записи на первую запись в файле (значение НФ) или в конец файла (значение КФ); значение по умолчанию — НФ. Диспозиция определяет состояние файла после обработки. Значение СОХР указывает, что файл после закрытия должен быть сохранен. Значение УНИЧ указывает, что после закрытия файл должен быть уничтожен. Значение по умолчанию — СОХР.

⟨срок хранения файла⟩ — указывает срок хранения в сутках. Значение по умолчанию — 7 суток.

⟨организация файла⟩ — допустимые значения: П — последовательная; К — индексно-последовательная; Б — библиотечная; Э70 — прямая (или блочная). Значение по умолчанию — Э70.

⟨формат записи⟩ — допустимые значения: Ф — если длина всех записей постоянная; П — если длина может изменяться от записи к записи. Значение по умолчанию — Ф.

Открытие файла:

ОТФ КРН = ⟨код района⟩
ФАЙ = ⟨имя файла⟩
КФЛ = ⟨код файла⟩

[ПОК = ⟨поколение⟩]
([РЕЖ = ⟨режим использования и обработки⟩])
([ДИС = ⟨статус, диспозиция⟩])
[СХР = ⟨срок хранения файла⟩]-

⟨поколение⟩ — указывает на создание очередного поколения файла либо на возврат к ранее созданному поколению. Если при создании очередного поколения их допустимое количество превышено, то будет уничтожено самое старое поколение.

⟨режим использования и обработки⟩ — списковый операнд. Режим использования может быть монопольным (значение М) или совместным (значение С). Значение по умолчанию — С. Подоперанд «режим обработки» может принимать значения: Ч — только чтение; З — чтение и запись; У — чтение, запись и уничтожение. Значение по умолчанию — Ч.

⟨срок хранения файла⟩ — служит для назначения нового срока хранения с момента исполнения процедуры.

Закрытие файла:

ЗКФ КФЛ = ⟨код файла⟩
[КЛУ = ⟨ключ уничтожения⟩]-

Операнд КЛУ задается в том случае, если при открытии файла задана диспозиция УНИЧ, а файл защищен ключом.

Уничтожение файла:

УНФ КРН = ⟨код района⟩
ФАЙ = ⟨имя файла⟩
[КЛУ = ⟨ключ уничтожения⟩]-

Файл будет уничтожен только в том случае, если он не открыт ни одной задачей.

Создание раздела:

СРФ КФЛ = ⟨код файла⟩
РАЗ = ⟨имя раздела⟩
КРЗ = ⟨код раздела⟩
([ПРВ = ⟨пространство под раздел⟩])
[ОРГ = ⟨организация раздела⟩]-

⟨имя раздела⟩ — последовательность от одного до двух простых имен, составляемая по тем же правилам, что и имя файла.

⟨код раздела⟩ — логический код, с помощью которого устанавливается принадлежность обрабатываемых записей данному разделу.

Поиск и открытие раздела:

НРФ КФЛ = ⟨код файла⟩
РАЗ = ⟨имя раздела⟩
КРЗ = ⟨код раздела⟩-

Режим обработки раздела определяется на основании режима обработки файла и полномочий пользователя, которыми он обладает при работе с данным разделом.

Закрытие раздела:

ЗРФ КФЛ = ⟨код файла⟩

РАЗ = ⟨имя раздела⟩⁻

Уничтожение раздела:

УРФ КФЛ = ⟨код файла⟩

РАЗ = ⟨имя раздела⟩⁻

Пространство, занимаемое разделом, освобождается лишь когда файл открыт с монопольным режимом использования.

Каждая процедура управления районами, файлами и разделами может привести к нескольким исходам:

— процедура выполнена полностью и нормально;

— обнаружены ошибки при синтаксическом анализе процедуры;

— обнаружена ошибка при выполнении процедуры;

— не разрешен доступ к файлу из-за его монопольного использования другой задачей.

При обнаружении ошибки в статически заданной процедуре на листинг распечатывается текст диагностического сообщения и сама процедура, при выполнении которой обнаружена ошибка.

§ 2. Экстракоды и их функции

В ходе решения задачи обращение к УПД ДИАПАК может быть выполнено по экстракоду 054. Исполнительный адрес экстракода и информация, заданная к нему на сумматоре, регистре младших разрядов или в информационном поле, определяют функцию экстракода.

Выполнение экстракода всегда завершается выдачей кода ответа от системы. Каждому значению кода ответа соответствует текст диагностического сообщения, который может быть распечатан на листинг задачи или передан в оперативную память задачи.

Экстракод *динамического задания процедур* предназначен для исполнения процедур управления районами, файлами и разделами в ходе решения задачи. При этом установка необходимых томов внешней памяти обеспечивается системой.

Если синхронизацию выполняет система, задача «закрывается» до полного выполнения процедуры. Значение кода ответа характеризует результат выполнения процедуры.

Если синхронизацию выполняет задача, она получит управление сразу же после передачи процедуры системе управления данными. Прием новых процедур от данной задачи система блокирует, пока не выполнит принятую процедуру. Значение кода ответа характеризует успешную или неуспешную передачу процедуры системе. О готовности результата выполнения принятой процедуры система информирует задачу по событию «выполнение процедуры

завершено». Результат выполнения процедуры может быть запрошен по экстракоду.

Экстракод *запроса результата выполнения процедуры* используется при наступлении события «выполнение процедуры завершено». Код ответа характеризует результат выполнения процедуры. Если экстракод выполнен преждевременно, то задача получит соответствующий код ответа.

Экстракод *протоколирования процедуры на листинг задачи* используется для распечатки тела процедуры и текста диагностического сообщения, соответствующего коду ответа.

Экстракод *выдачи диагностического сообщения в память задачи* используется, как правило, в тех случаях, когда необходимо протоколировать результат обращения к УПД ДИАПАК на какое-либо специальное устройство вывода (например, на терминал).

Экстракод *обеспечения файла пространством для размещения данных*, как правило, выполняется системой автоматически на операциях обработки записей. Но при этом может возникнуть ситуация, когда система «закроет» задачу на продолжительное время. Например, в тех случаях, когда свободного пространства не оказалось, и необходимо выполнить разгрузку района или пула. Если такие «закрытия» задачи нежелательны, необходимо выполнить данный экстракод.

Если получен ответ — «заказ принят, жди событие» и тем не менее задача обратилась к файлу по экстракоду обработки записей до наступления события, то выполнение синхронизации система берет на себя и «закрывает» задачу.

Экстракод *опроса состояния заказов, переданных системе*, необходим для получения информации о невыполненных заказах.

Экстракод *запроса свободных кодов файлов*. Свободные коды файлов определяются с учетом как открытых файлов, так и занятых логических номеров магнитных лент и дисков к моменту выполнения экстракода.

Экстракод *запроса сведений об открытом файле*. По экстракоду выдается в память задачи информация об организации файла, режима обработки, используемом пространстве и т. п.

Экстракод *коррекции номера последней зоны файла*. Последняя используемая зона файла отслеживается системой автоматически. По номеру этой зоны файл обеспечивается пространством. Если известно, что, начиная с некоторой зоны, хранить информацию не нужно (т. е. не имеет смысла занимать пространство в районе), об этом следует сообщить системе по этому экстракоду.

Если обработка файла осуществляется в пуле, то данный экстракод позволяет отказаться от изменений, внесенных в файл, и запросить исходную версию файла из района при очередном открытии.

§ 3. Операции обработки записей

Для файла или раздела с прямой организацией передача данных осуществляется с помощью экстракода 070. Логический номер устройства, указанный в информационном слове, интерпретируется как код файла, номер зоны — как номер записи. Запись имеет фиксированную длину 1024 слова и может быть обработана только с начала листа.

Для файлов и разделов с последовательной и индексно-последовательной организацией обработка данных (чтение, запись, уничтожение) осуществляется по экстракоду 054. Запись, как минимальный элемент таких файлов, характеризуется длиной и именем. Длина записи ограничена лишь размером оперативной памяти, которой располагает программа обработки файла. Именем записи служит порядковый номер записи в последовательном файле и ключ записи в индексно-последовательном файле.

При включении новой записи в последовательный файл система присваивает ей порядковый номер и сообщает его пользователю. При включении записи в индексно-последовательный файл пользователь должен присвоить ей ключ. Имена записей (номера, ключи) могут быть использованы для поиска конкретной записи.

Передача записи в файл и выборка ее из файла могут быть выполнены с любого адреса оперативной памяти задачи. Операции обработки записей допускают выборку или модификацию части записи. Обращение к записям файла может быть выполнено с использованием как их уникальных имен, так и отношения следования, установленного при формировании файла (по возрастанию значений имен записей). В последнем случае имена обрабатываемых записей могут быть дополнительно затребованы задачей. В одной операции выборки или уничтожения записей возможна обработка группы записей в логическом порядке следования.

При последовательной обработке файла используется указатель на текущую запись («стрелка»). Первоначальная установка «стрелки» происходит при открытии файла операндом СТАТУС. После выполнения очередной операции обработки «стрелка» устанавливается на запись, следующую по возрастанию имен за последней обработанной записью. Если обращение к записи выполнено по имени и искомой записи нет, «стрелка» устанавливается на ближайшую запись с большим именем.

Весь набор операций обработки записей разбит на 4 группы:

— внесение записей в файл;

— выборка записей в оперативную память задачи;

— уничтожение записей;

— обновление файла.

Внесение записей в файл. УПД ДИАПАК допускает следующие модификации этой операции:

1) Внесение новой записи в файл. Для последовательной организации запись принимается в конец файла и задаче сообщается порядковый номер записи в файле. Для индексно-последовательной организации запись принимается в файл вместе с ее именем (ключом) и включается в порядке возрастания значения ключа (ключ рассматривается как логическая константа). Если запись с данным ключом уже существует, то выдается сообщение об ошибке.

2) Безусловная передача записи в файл по имени. Если запись с указанным именем существует, то она уничтожается, а на ее место принимается новая запись. Причем для последовательного файла, если размер записи не увеличился, она принимается на старое место и за ней сохраняется порядковый номер в файле, иначе запись принимается в конец файла и задаче сообщается ее новый порядковый номер.

3) Безусловная передача записи в последовательный файл по указателю на текущую запись.

4) Замена записей в файле по имени. Если запись с указанным именем не существует либо размер записи (для последовательного файла) не увеличился, то выдается сообщение об ошибке.

5) Замена записи в последовательном файле по указателю на текущую запись.

6) Дозапись. Массив, указанный в экстракоде, приписывается в конец уже существующей записи. В последовательном файле дозапись осуществляется к последней записи. В индексно-последовательном файле может быть увеличена любая запись.

Выборка записей из файла. УПД ДИАПАК допускает следующие модификации этой операции:

1) Выборка записи по имени. Если записи с данным именем нет, то выдается ошибка.

2) Безусловная выборка записи по имени. Если записи с данным именем нет, то первой выбирается запись с ближайшим бо́льшим именем к заданному.

3) Выборка записи по указателю на текущую запись.

4) Выборка записи по указателю на текущую запись с выдачей имени записи.

Уничтожение записей. УПД ДИАПАК допускает следующие модификации этой операции:

1) Уничтожение записи по имени. Если записи с данным именем нет, то выдается ошибка.

2) Безусловное уничтожение записи по имени. Если записи с данным именем нет, то первой будет уничтожена запись с ближайшим по значению именем к заданному.

3) Уничтожение записи по указателю на текущую запись.

4) Уничтожение записи по указателю на текущую запись с выдачей имен уничтоженных записей.

Обновление файла. Предназначена для уничтожения в нем всех записей или части файла. Если операция обновления применяется к последовательному файлу, то указывается номер записи, начиная с которой необходимо уничтожить все записи до конца файла. Если операция обновления применяется к индексно-последовательному файлу, то осуществляется уничтожение всех записей в файле.

Выполнение операций обработки записей сопровождается выдачей ответа. Система выдает положительный ответ тогда и только тогда, когда переданная по экстракоду заявка выполнена полностью. Если ответ отрицательный, то либо заявка составлена некорректно, либо заявка не может быть выполнена в полном объеме, либо обнаружен системный отказ.

§ 4. Замечания по использованию языка взаимодействия

Прежде чем использовать операции управления файлами, необходимо открыть район, с файлами которого пользователь предполагает работать. Одновременно могут быть открыты максимум 31 район. Причем все коды районов должны быть различными. Разрешается открывать один и тот же район дважды и более с различными кодами районов.

Прежде чем использовать операции управления разделами библиотечного файла, необходимо открыть этот файл.

Если в паспорте определяются (или открываются) два или более файлов (или разделов) с одинаковыми кодами файлов (или кодами разделов), то воспринимается последняя из этих операций (все предшествующие игнорируются). В динамике такое совпадение приводит к ошибке «повторяется код файла (код раздела)».

Разрешается открывать дважды и более один и тот же файл (или раздел) с разными значениями кода файла (раздела), но режимы использования этого файла должны быть только «совместно».

Разрешается одновременно иметь открытыми до 32 файлов.

По окончании задачи система автоматически закрывает районы, файлы и разделы, если задача «не успела» их закрыть с помощью соответствующих операций. При вызове операционной системы после сбоя также автоматически закрываются все районы, файлы и разделы, открытые до сбоя.

СПИСОК ЛИТЕРАТУРЫ

1. Самарский А. А. Пакеты прикладных программ как средства обеспечения сложных физических расчетов.— В кн.: Перспективы системного и теоретического программирования. Труды Всесоюзного семинара. Новосибирск: ВЦ СО АН СССР, 1979, с. 5—14.
2. Яненко Н. Н., Карначук В. И., Коновалов А. Н. Проблемы математической технологии.— В кн.: Численные методы механики сплошной среды. Новосибирск, 1977, 8, № 3, с. 129—157.
3. Горбунов-Посадов М. М., Карпов В. Я., Корягин Д. А., Красотченко В. В., Самарский А. А. Об одном подходе к автоматизации программирования вычислительного эксперимента.— В кн.: Структура и организация пакетов программ. Тезисы докладов. Тбилиси: Мецниереба, 1976, с. 27—28.
4. Карпов В. Я., Корягин Д. А., Самарский А. А. Принципы разработки пакетов прикладных программ для задач математической физики.— М., 1977. (Препринт/ИПМ АН СССР: № 86.)
5. Воронков А. В., Легоньков В. И. О системе программного обеспечения больших задач математической физики.— М., 1979. (Препринт/ИПМ АН СССР: № 45.)
6. Легоньков В. И., Петров А. А. Некоторые общие вопросы разработки и эксплуатации больших программ для счета задач математической физики.— В кн.: Комплексы программ математической физики. Новосибирск: ВЦ СО АН СССР, 1972, с. 58—64.
7. Софронов И. Д. Оценка параметров вычислительной машины, предназначенной для решения задач механики сплошной среды.— В кн.: Численные методы механики сплошной среды. Новосибирск, 1976, 6, № 3.
8. Шалфеев А. Д. Выбор функциональной структуры и исследование средств, обеспечивающих режим общих ресурсов в рамках многомашинного вычислительного комплекса.— Вопросы атомной науки и техники. Сер. Методики и программы численного решения задач математической физики. М.: ЦНИИатоминформ, 1979, вып. 3(5), с. 23—36.
9. Мазурин Ю. Н., Шалфеев А. Д., Золотилин А. К., Петрушко Н. Н., Папков М. А. Аппаратура коммутации внешней памяти и вычислительных машин.— В кн.: Новые средства аппаратного обеспечения БЭСМ-6. М.: ВЦ АН СССР, 1976, с. 61—65.

10. Лузганов В. П., Кибардина Н. П., Русак Г. Г., Туманов В. А. Аппаратура сопряжения терминалов (типа VIDEOTON-340, Т-63) с ЭВМ БЭСМ-6.— В кн.: Новые средства аппаратного обеспечения БЭСМ-6. М.: ВЦ АН СССР, 1976, с. 20—28.
11. Шулепов Н. И. Диалого-пакетная операционная система для многомашинного комплекса ЭВМ БЭСМ-6 (ОС ДИАПАК).— Вопросы атомной науки и техники. Сер. Методики и программы численного решения задач математической физики. М.: ЦНИИатоминформ, 1979, вып. 1(3), с. 12—19.
12. Мазурин Ю. Н., Озорнин Ю. В., Охрименко Г. П., Шулепов Н. И. Базовый уровень программного обеспечения общей дисковой памяти многомашинного комплекса в ОС ДИАПАК.— Вопросы атомной науки и техники. Сер. Методики и программы численного решения задач математической физики. М.: ЦНИИатоминформ, 1979, вып. 1(3), с. 20—32.
13. Опарин А. А., Бартенев Ю. Г., Кривов Е. С. Программные средства, предоставляемые пользователям для организации параллельных вычислений на системе машин БЭСМ-6.— Вопросы атомной науки и техники. Сер. Методики и программы численного решения задач математической физики. М.: ЦНИИатоминформ, 1979, вып. 3(5), с. 15—22.
14. Давыдов Ю. М. Об опыте использования дисплейной техники при проведении численного эксперимента.— В кн.: Труды III семинара по комплексам программ математической физики. Новосибирск: ВЦ СО АН СССР, 1973, с. 48—57.
15. Корякин В. К., Соколов В. П. Символьный отладчик в ОС ДИАПАК.— Вопросы атомной науки и техники. Сер. Методики и программы численного решения задач математической физики. М.: ЦНИИатоминформ, 1979, вып. 1(3), с. 33—37.
16. Селиванов И. В., Забабахина Н. В., Шевченко Т. П., Украинская Т. В. Мониторная система ДУБНА в ОС ДИАПАК. Диалоговый режим.— Свердловск: ИММ УНЦ АН СССР, 1981.
17. Королев Л. Н. Структуры ЭВМ и их математическое обеспечение.— М.: Наука, 1978.
18. Информационные системы общего назначения/Пер. с англ. под ред. Е. Л. Ющенко.— М.: Статистика, 1976.
19. Бушев С. Н., Бондаренко В. И. Банки данных (обзор).— Зарубежная радиоэлектроника, 1974, № 7, с. 3—23.
20. Мэдник С., Донован Дж. Операционные системы/Пер. с англ. под ред. Л. Д. Райкова.— М.: Мир, 1978.
21. Катцан Г. Операционные системы/Пер. с англ. под ред. К. А. Пупкова и Л. И. Шатровского.— М.: Мир, 1976.
22. Последов Г. В., Райков Л. Д. Введение в ОС ЕС ЭВМ.— М.: Статистика, 1977.
23. Уилкс М. Системы с разделением времени/Пер. с англ. под ред. Э. З. Любимского.— М.: Мир, 1972.
24. Горбунов-Посадов М. М., Хиздер Л. А. Работа с регионами и файлами в системе ОХРА.— М., 1979. (Препринт/ИПМ АН СССР: № 81.)
25. Ломтадзе В. В., Наумов В. А. Система оперирования данными для БЭСМ-6 (СОД/БЭСМ-6): Общее описание.— Иркутск: СЭИ СО АН СССР, 1978.
26. Столяров Г. К. Эволюция систем управления данными.— В кн.: Перспективы системного и теоретического программи-

рования. Труды Всесоюзного симпозиума. Новосибирск: ВЦ СО АН СССР, 1979.
27. Операционная система OS/360. Супервизор и управление данными/Пер. с англ. под ред. А. И. Илюшина.— М.: Советское радио, 1973.
28. Битти А. Архив.— В кн.: Супервизоры и операционные системы. Сер. Библиотека кибернетического сборника/Пер. с англ. под ред. В. С. Штаркмана.— М.: Мир, 1972, с. 97—105.
29. Единая система электронных вычислительных машин. Операционная система. Управление данными: Руководство программиста.— Ц51.804.005.14, 1980.
30. Илюшин А. И., Ерохов А. Н., Хованская Г. П. Управление данными для ОС ДИСПАК УПД-6. Ч. 1. Принципы построения системы.— М.: ИПМ АН СССР, 1974.
31. Илюшин А. И., Ерохов А. Н., Хованская Г. П. Управление данными для ОС ДИСПАК УПД-6. Ч. 2. Процедуры работы с данными и средства, предоставляемые пользователю.— М.: ИПМ АН СССР, 1974.
32. Тарасенко Л. Г., Янцен В. И. Файлы и архив в операционной системе.— М., 1970. (Препринт/ИТМ и ВТ АН СССР: № 16.)
33. Курляндчик Я. М. Система управления файлами.—М., 1980. (Препринт/ИТМ и ВТ АН СССР: № 13.)
34. Веретенов В. Ю., Волков А. И., Гуревич М. И., Козик В. С., Подъячев Е. И., Шапиро М. Л. Дисковая операционная система.— М., 1975. (Препринт/ИА: № 2486.)
35. Wimmer W. Uber die Massenspeicherhierarchie als Teil eines Datenverwaltungssystems am DES. Y.— Rechenzentrum.— Angew. Inform. 1978, 20, № 9, p. 381—388.
36. Spitz E., Guval A. Public online pool — a method for managing online direct access space.— Soft — ware — Pract. and Exper., 1979, 9, № 2, p. 139—147.
37. Мартин Дж. Организация баз данных в вычислительных системах/Пер. с англ. под ред. А. Л. Щерса.— М.: Мир, 1978.
38. Бертэн Ж., Риту М., Ружие Ж. Работа ЭВМ с разделением времени/Пер. с англ. под ред. С. С. Лаврова.— М.: Наука, 1972.
39. Катцан Г. Вычислительные машины и системы 370/Пер. с англ. под ред. В. К. Левина и Л. Д. Райкова.— М.: Мир, 1974.
40. Цикритзис Д., Бернстайн Ф. Операционные системы/Пер. с англ. под ред. И. Б. Задыхайло и В. В. Мартынюка.— М.: Мир, 1977.
41. Дейт К. Введение в системы баз данных/Пер. с англ.— М.: Наука, 1980.
42. Системы банков данных. Аналитический обзор по материалам фирмы IBM.— М.: ВИМИ, 1975.
43. СМО «СИНБАД-2». Руководство системного программиста.— Калинин: НПО «Центрпрограммсистем», 1977.
44. Единая система электронных вычислительных машин. Система управления базами данных ОКА: Общее описание.— М.: Госплан СССР, 1976.
45. Otte R. Die Anwendunge eines Datenbanksystems für die Magnetenbandarchivierung. — Rechentechnik/Datenverarbeitung, 1976, 13, № 3, p. 37—40.

46. Буглаева Л. Д., Крылов Г. М., Лубовина Л. М. Учет использования томов и контроль доступа к информации.— Вопросы радиоэлектроники. Сер. Электронная вычислительная техника, 1975, вып. 5, с. 54—64.
47. Krah P., Bittner J. Speicherstruktur und Sprachkonzept der DBS/R.— Rechentechnik/Datenverarbeitung, 1977, 14, Beih, № 4, p. 16—29.
48. Winspur W. A design for file management in a computer network.— Proceedig of the Eight Hawell Int. Conf. on Syst. Sci. West. Periodic. Co., 1975, p. 126—128.
49. Бурков В. Н., Соколов В. Б. Оптимальное размещение информационных массивов в памяти на магнитных лентах для случая двунаправленного поиска.— Автоматика и телемеханика, 1969, № 4, с. 107—117.
50. Литвинов В. А. Алгоритмы оптимизации размещения массивов информации на магнитной ленте.— Кибернетика, 1970, № 4.
51. Матер Е. А., Филина Н. Е. Оптимизация размещения массивов на магнитной ленте в автоматизированных системах управления.— Кибернетика, 1971, № 6, с. 130—139.
52. Бурков В. Н., Рубинштейн М. И., Соколов В. Б. Некоторые задачи оптимального размещения информации в памяти большого объема.— Автоматика и телемеханика, 1969, № 9, с. 83—91.
53. Гендель Е. Г., Подвин З. Е. Оптимальное распределение оперативного накопителя ЭВМ при обработке больших массивов информации.— В кн.: Автоматизированные системы управления. Минск: ЦНИИТУ, 1971, вып. 5.
54. Шеперов Н. А. Одна задача оптимального распределения ресурсов одноуровневой внешней памяти.— Управляющие системы и машины, 1978, № 5, с. 19—22.
55. Носаль Г. В., Репкина Л. К. Автоматическое распределение двухуровневой памяти в системе с разделением времени.— Вопросы радиоэлектроники. Сер. Электронная вычислительная техника, 1976, вып. 8, с. 58—67.
56. Гостев Ю. Г., Левочкин Е. А., Леонтьев А. С., Наголкин А. Н. Организация хранения информации в единой системе автоматизации проектирования.— Вопросы радиоэлектроники. Сер. Электронная вычислительная техника, 1978, вып. 1, с. 81—88.
57. Кутепов В. П. Активное множество страниц программы и его поведение.— Программирование, 1975, № 1, с. 14—21.
58. Стоян Ю. А. Результаты оценки оптимального алгоритма замещения.— Программирование, 1975, № 2.
59. Ерохов А. Н., Илюшин А. И,. Клепиков Н. Ю. Управление данными УПД-6. Организация локального банка данных. Структура ЛБД.— М., 1976. (Препринт/ИПМ АН СССР: № 121.)
60. Ерохов А. Н., Илюшин А. И., Клепиков Н. Ю. Управление данными УПД-6. Обслуживание локального банка данных: Руководство для администратора.— М., 1977. (Препринт/ИПМ АН СССР: № 117.)
61. Крюков В. М. Управление данными в УПД ДИАПАК.— М., 1981. (Препринт/ИПМ АН СССР: № 123.)
62. Зуев В. И., Крюков В. М. Система управления данными в ОС ДИАПАК.— Вопросы атомной науки и техники. Сер. Ме-

тодики и программы численного решения задач математической физики. М.: ЦНИИатоминформ, 1981, 1(7), с. 3—8.
63. Зуев В. И., Крюков В. М., Озорнин Ю. В., Шулепов Н. И. Функциональная структура системы АРХИВ.— В кн.: Комплексы программ математической физики. Материалы 5-го Всесоюзного семинара по комплексам программ математической физики. Новосибирск: ИТ и ПМ СО АН СССР, 1978, с. 171—172.
64. Зуев В. И., Крюков В. М., Озорнин Ю. В., Шулепов Н. И. Технология обработки данных в системе АРХИВ.— В кн.: Комплексы программ математической физики. Материалы 5-го Всесоюзного семинара по комплексам программ математической физики. Новосибирск: ИТ и ПМ СО АН СССР, 1978, с. 173—174.
65. Зуев В. И. Обеспечение устойчивости функционирования системы управления данными УПД ДИАПАК.— М., 1981. (Препринт/ИПМ АН СССР: № 124.)
66. Зуев В. И., Крюков В. М., Попова Т. В., Романова Е. М. Управление данными в ОС ДИАПАК. Возможности и средства, предоставляемые пользователю.— Свердловск: ИММ УНЦ АН СССР, 1981.
67. Зуев В. И., Козырева Е. В., Крюков В. М., Романова Е. М. Система управления данными УПД ДИАПАК. Унифицированный пакет обслуживающих программ: Руководство пользователю.— Свердловск: ИММ УНЦ АН СССР, 1981.
68. Зуев В. И., Козырева Е. В., Крюков В. М. Управление данными в ОС ДИАПАК: Руководство администратору данных.— Свердловск: ИММ УНЦ АН СССР, 1981.
69. Элмендорф В. Р. Упорядоченная проверка математического обеспечения.— В кн.: Средства отладки больших систем/Пер. с англ.— М.: Статистика, 1977, с. 125—127.
70. Липаев В. В. Проблемы обеспечения надежности и устойчивости функционирования комплексов АСУ.— Управляющие системы и машины, 1977, № 3, с. 39—45.
71. Головкин Б. А. Надежное программное обеспечение: Обзор.— Зарубежн. радиоэлектрон., 1978, № 12, с. 3—61.
72. Майерс Г. Надежность программного обеспечения/Пер. с англ. под ред. В. Ш. Кауфмана.— М.: Мир, 1980.
73. Липаев В. В. Надежность программного обеспечения АСУ.— М.: Энергоиздат, 1981.
74. Шураков В. В. Надежность программного обеспечения систем обработки данных.— М.: Статистика, 1981.
75. Дал У., Дейкстра Э., Хоор К. Структурное программирование/Пер. с англ. под ред. Э. З. Любимского и В. В. Мартынюка.— М.: Мир, 1975.
76. Кинг Дж. Проверяющий компилятор.— В кн.: Средства отладки больших систем/Пер. с англ.— М.: Статистика, 1977.
77. Половко А. М. Основы теории надежности.— М.: Наука, 1964.
78. Журавлев Ю. П., Котелюк Л. А., Циклинский Н. И. Надежность и контроль ЭВМ.— М.: Советское радио, 1978.
79. Randell B. System structure for software fault tolerance.— IEEE Transaction on Software Engineering, 1975, vol. Se — 1, № 2, p. 220—232.

80. Hecht H. Fault-tolerant software for real-time applications.— Computing Surveys, 1976, 8, № 4, p. 391—407.
81. Бокова И. Д., Зельдинова С. А., Зуев В. И., Корякин В. К., Кошкина Л. В., Озорнин Ю. В., Охрименко Г. П., Тюрин В. Ф., Шулепов Н. И. Операционная система ДИСПАК для БЭСМ-6 (пользователю).— М.: ИПМ АН СССР, 1973.
82. Зуев В. И., Корякин В. К., Кошкина Л. В., Крюков В. М., Озорнин Ю. В., Охрименко Г. П., Шулепов Н. И. Операционная система ДИАПАК: Руководство пользователю.— Свердловск: ИММ УНЦ АН СССР, 1981.
83. Базовский И. Надежность. Теория и практика/Пер. с англ. под ред. Б. Р. Левина.— М.: Мир, 1965.
84. Бобровников И. Д., Сорокина А. А. Функциональная структура операционных систем IBM/370 с виртуальной памятью: Обзор.— Зарубежн. радиоэлектрон., 1974, № 9, с. 3—21.
85. Справочник по цифровой вычислительной технике. Электронные вычислительные машины и системы/Под ред. Б. Н. Малиновского.— Киев: Техніка, 1980.
86. Гнеденко Б. В., Беляев Ю. К., Соловьев А. Д. Математические методы в теории надежности.— М.: Наука, 1965.
87. Яненко Н. Н. Проблемы математической технологии.— Структура и организация пакетов программ. Тезисы докладов. Тбилиси: Мецниереба, 1976, с. 9—11.
88. Ершов А. П. Некоторые субъективные замечания к актуальным проблемам программирования.— В кн.: Перспективы системного и теоретического программирования. Труды Всесоюзного симпозиума. Новосибирск: ВЦ СО АН СССР, 1979, с. 113—127.
89. Мясников В. А. Совершенствование технологии программирования — важнейшая народно-хозяйственная задача.— Управляющие системы и машины, 1980, № 1, с. 6—8.
90. Гвардейцев М. И., Морозов В. П., Розенберг В. Я. Специальное математическое обеспечение управления.— М.: Советское радио, 1980.
91. Надежность в технике. Термины и определения. Гост 13377—75.— М.: Издательство стандартов, 1975.
92. Байцер Б. Архитектура вычислительных комплексов. Том 2/Пер. с англ. под ред. Ю. П. Селиванова и Б. А. Квасова.— М.: Мир, 1974.
93. Единая система электронных вычислительных машин. Операционная система. Дополнительные возможности управления данными: Руководство системного программиста.— Ц51.804.006.Д103, 1980.
94. Единая система электронных вычислительных машин. Операционная система. Утилиты. Обслуживание системных управляющих данных и разметка магнитных лент: Руководство программиста.— Ц51.804.002.Д13, 1980.
95. Ерохов А. И., Илюшин А. И. Управление данными УПД-6. Дополнительные макрокоманды и утилиты.— М., 1978. (Препринт/ИПМ АН СССР: № 138.)
96. Ерохов А. Н. Управление данными УПД-6. Версия 5.0 локального банка данных. Утилита РЕВИЗ: Руководство для администратора. М.: ИПМ АН СССР, 1980.

97. Ерохов А. Н. Управление данными УПД-6. Версия 5.0 локального банка данных: Руководство администратора.— М.: ИПМ АН СССР, 1980.
98. Единая система электронных вычислительных машин. Операционная система. Программы обслуживания наборов данных: Руководство программиста.— Ц51.804.005.Д95, 1980.
99. Единая система электронных вычислительных машин. Операционная система. Макрокоманды управления данными: Руководство программиста.— Ц51.804.006.Д102, 1980.
100. Бокова И. Д., Зельдинова С. А., Зуев В. И., Корякин В. К., Кошкина Л. В., Озорнин Ю. В., Охрименко Г. П., Тюрин В. Ф., Шулепов Н. И. Операционная система ДИСПАК для БЭСМ-6 (системному программисту и оператору).— М.: ИПМ АН СССР, 1973.
101. Михеев В. М., Штаркман В. С. Макрокод: Описание языка.— М., 1972. (Препринт/ИПМ АН СССР: № 24.)
102. Михеев В. М., Вершубский В. Ю. АСТРА. Язык для записи алгоритмов системного программирования и трансляции.— М., 1974. (Препринт/ИПМ АН СССР: № 16.)
103. Единая система электронных вычислительных машин. Операционная система. Средства восстановления в операционной системе ОС ЕС: Руководство системного программиста.— Ц51,804.006.Д21, 1980.
104. Единая система электронных вычислительных машин. Операционная система. Средства восстановлений системы ввода-вывода: Руководство системного программиста.— Ц51.804.006.Д66, 1980.
105. Основы теории вычислительных систем/Под ред. С. А. Майорова.—М.: Высш. шк., 1978.
106. Бусленко Н. П., Калашников В. В., Коваленко И. Н. Лекции по теории сложных систем.— М.: Сов. радио, 1973.
107. Теория надежности и массовое обслуживание.— М:. Наука, 1969.
108. Бунатян А. А., Легоньков В. И., Мазурин Ю. Н., Озорнин Ю. В., Шалфеев А. Д., Шулепов Н. И. Операционная система многомашинного комплекса ЭВМ БЭСМ-6, работающего в режиме разделения времени (ОС ДИАПАК).— В кн.: Автоматизация проектирования систем автоматического и автоматизированного управления. 2-е Всесоюзное научно-техническое совещание. Тезисы докладов. Челябинск, 1978, с. 16—18.
109. Davenport R. A. Distributed or centralised datebase — a methology for selection.— Convention informatique. Paris, 1976, p. 76—83.
110. Ерохов А. Н., Илюшин А. И. Соглашение о связях между программой и подпрограммой.— М., 1974. (Препринт/АН СССР: № 120.)
111. Shneiderman B. Jump searching: a fast sequential search technique.— Communs. ACM, 1978, 21, № 10, p. 831—834.
112. Золотилин А. К., Лузганов В. П., Мазурин Ю. Н., Онищук Н. Н., Папков М. А., Шалфеев А. Д. Практические вопросы работы и совершенствования вычислительного комплекса. Материалы 6 конференции по БЭСМ-6.— Тбилиси: ИПМ ТГУ, 1977.

СПИСОК СОКРАЩЕНИЙ

В данный список внесены неоднократно встречающиеся в тексте русские и английские сокращенные обозначения терминов и слов. Список имеет два раздела: сокращения русских наименований и сокращения английских наименований.

АП — анализатор процедур

АСТРА — язык для записи алгоритмов системного программирования и трансляции

АЦПУ — алфавитно-цифровое печатающее устройство

ВП — внешняя память

ВРТ — подсистема восстановления резидентного тома

ВС — вычислительная система

ВЦ — вычислительный центр

ДРТ — дубль резидентного тома

КРП — каталог районов и пулов

КСИ — подсистема контроля служебной информации

КФ — каталог файлов

ЛБД — локальный банк данных

МБ — магнитный барабан

МВК — многомашинный вычислительный комплекс

МД — магнитный диск

МЛ — магнитная лента

ММ — метка множества

МП — метка пула

МР — метка района

МРВП — модуль распределения внешней памяти

МРРФ — модуль рационального размещения файлов

МТ — метка тома

МУМ — модуль управления множествами

МУФ — модуль управления файлами

МФ — метка файла

ОКА — система управления базами данных ОКА

ОП — оперативная память

ОС — операционная система

ОС ДИАПАК — диалого-пакетная операционная система для многомашинного комплекса ЭВМ БЭСМ-6

ОС ДИСПАК — операционная система пакетной обработки для ЭВМ БЭСМ-6

ОС ЕС — операционная система единой системы электронных вычислительных машин

ПМД — программы методов доступа

РТ — резидентный том
СИ — служебная информация
СИНБАД-2 — система интегрированных баз данных СИНБАД-2
СУБД — система управления базами данных
СУД — система управления данными
ТОФ — таблица открытых файлов
ТРПФ — таблица распределения пространства под файлы
ТСЗ — таблица свободных зон
УНИПАК — унифицированный пакет обслуживающих программ
УП — управляющая программа
УПД-6 — система управления данными для ОС ДИСПАК
УПД ДИАПАК — система управления данными в ОС ДИАПАК
УПД-ЕС — управление данными в ОС ЕС
DBS/R — система банка данных DBS/R
DD — предложение описания данных
DCB — таблица управления данными
IBM/360-168 — электронная вычислительная машина фирмы IBM, семейства 360, модель 168
IMS — система управления информацией
MULTICS — операционная система Массачусетского технологического института
OS/360 — операционная система для семейства 360
TSS/360 — система разделения времени для семейства 360

Виталий Иванович Зуев
Вячеслав Михайлович Крюков
Владимир Иванович Легоньков

УПРАВЛЕНИЕ ДАННЫМИ
В ВЫЧИСЛИТЕЛЬНОМ ЭКСПЕРИМЕНТЕ

Серия: «Библиотечка программиста»

Редактор *Н. И. Воронина*
Художественный редактор *Т. Н. Кольченко*
Технический редактор *С. Я. Шкляр*
Корректоры *Г. В. Подвольская, Л. С. Сомова*

ИБ № 12831

Сдано в набор 14.08.85. Подписано к печати 29.05.86. Т-14030. Формат 84×108$^{1}/_{32}$. Бумага тип. № 3. Гарнитура обыкновенная. Печать высокая. Усл. печ. л. 8.4. Усл. кр.-отт. 8,61. Уч.-изд. л. 9,46. Тираж 17 000 экз. Заказ № 861. Цена 55 коп.

Ордена Трудового Красного Знамени
издательство «Наука»
Главная редакция физико-математической литературы
117071 Москва В-71, Ленинский проспект, 15

4-я типография издательства «Наука»
630077 Новосибирск, 77, Станиславского, 25

ИЗДАТЕЛЬСТВО «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
117071 Москва В-71, Ленинский проспект, 15

ГОТОВИТСЯ К ПЕЧАТИ

Кушниренко А. Г., Лебедев Г. В. **Практическое программирование:** Учеб. пособие.

Излагается курс практического программирования для вузов. Основу книги составляют технология программирования «сверху — вниз» и развитые структуры данных.

Курс рассчитан на решение большого количества задач по изучению и модификации тщательно подготовленных эталонных текстов программ. В числе этих программ — реализация различных структур данных, реализация мини-проектов, а также около 100 простых упражнений.

Изложение ведется на учебно-производственном языке высокого уровня. Приводятся правила и примеры перекодировки программ с этого языка на язык фортран.

Для студентов университетов и факультетов прикладной математики.

Предварительные заказы принимаются без ограничений магазинами Книготорга и Академкниги.

ИЗДАТЕЛЬСТВО «НАУКА»
ГЛАВНАЯ РЕДАКЦИЯ
ФИЗИКО-МАТЕМАТИЧЕСКОЙ ЛИТЕРАТУРЫ
117071 Москва В-71, Ленинский проспект, 15

ГОТОВИТСЯ К ПЕЧАТИ

Глушков В. М. **Основы безбумажной информатики.— 2-е изд., испр.**

Представлено комплексное систематизированное изложение современного состояния и перспектив информатики и связанных с нею научно-технических проблем. Математические методы излагаются без доказательств.

Отражены проблемы автоматизированной обработки и хранения информации в машинном (безбумажном) представлении, т. е. при помощи ЭВМ.

Для инженеров и специалистов в различных отраслях народного хозяйства, студентов, а также для желающих познакомиться с основами и перспективами информатики.

Первое издание (1982 г.) было встречено с большим интересом специалистами различного профиля, связанными в своей работе с применением ЭВМ.

Предварительные заказы принимаются без ограничений магазинами Книготорга и Академкниги.