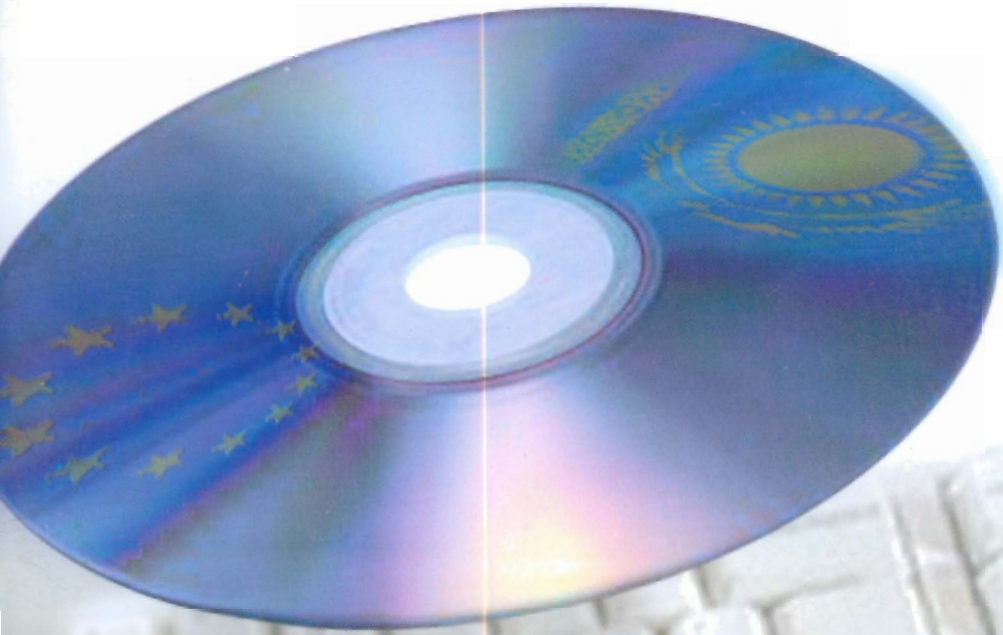
Г. Балакаева

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Operating Systems



Учебное пособие по
Computer Science

TEMPUS TACIS
Contract # CD JEP 22077-2001

Казахский национальный университет им. аль-Фараби
Научно-исследовательский институт математики и механики
Механико-математический факультет
Кафедра информатики

Г. Балакаева

Операционные системы
Operating systems

Учебное пособие по *Computer Science*
Образовательная программа Европейского Союза
TEMPUS-TACIS (*Contract # CD_JEP*-22077-2001)

Алматы 2004
Издательство "*Print S*"

УДК 004.451
ББК 32.97я 7
Б 20

TEMPUS TACIS Project
"Development of Curriculum for Master Degree Programme in Computer Science and Computer-based Learning in the Institute of Mechanics and Mathematics of al-Farabi Kazakh National University"
Contract # CD_JEP-22077-2001

Рецензенты:
профессора J. Vila (*Universidad Politecnica de Valencia*, Испания) и
С. Я. Серовайский (Казахский национальный университет им. аль-Фараби, Казахстан).

Балакаева Г.
Б 20. **Операционные системы:** Учебное пособие. – Алматы: Издательство ТОО "*Print S*", 2004. – 59 с.

ISBN 9965-9343-6-3

Учебное пособие написано в рамках образовательного проекта Европейского Союза по программе *TEMPUS TACIS* (*Contract # CD_JEP-22077-2001*). Целью проекта являлась разработка учебного плана для магистратуры по специальности "*Информатика*" в Казахском национальном университете имени аль-Фараби. В работе над проектом принимали участие также *University of Newcastle upon Tyne* (Ньюкасл, Великобритания), *Universidad Politecnica de Valencia* (Валенсия, Испания) и *Blekinge Institute of Technology* (Роннеби, Швеция).

Пособие может быть рекомендовано студентам информационных, физико-математических и технических и прикладными аспектами информатики и компьютерных специальностей, а также аспирантам, научным работникам, инженерам и специалистам, интересующимся теоретическими технологий.

УДК 004. 451
ББК 32.97я 7

Б $\frac{2404010000}{00(05)-04}$

ISBN 9965-9343-6-3

Содержание

Preface

Operating systems are one of the more challenging fields of computing. There are many reasons for that. From the user point of view, they provide the interface and the basic set of concepts to work with a computer, making it a useful tool for running many different kinds of applications. Using computers requires to minimally understand operating system abstractions like "files", "processes", "drivers" and so on. As the use of computers spread to all aspects of modern life, all these concepts are more and more present in the work, the leisure and, in general, the daily life of a growing number of people. The impact of operating systems on economy is also patent by the fact that they represent the major software business in the world. But, at the same time, they also represent the largest free software project: Linux, where thousands of students and researchers all over the world collaborate in the construction of a powerful operating system.

Operating systems play also a central role in computer education. They are considered to be a core subject in the computer science curricula of all universities and in the curricula recommendations of all prestigious research and professional associations, like IEEE or ACM. The reason is that operating systems provide a whole understanding of the computer functioning, both hardware and software. Operating systems could be considered in this way as the "art" of managing hardware resources and creating an idealised (abstract) view of the same so as to facilitate users their use and also sharing those resources with other users. Operating systems are also in the origin of new disciplines of computing. This way the book by F. P. Brooks "*The mythical man month. Essays on software engineering*" describing their experiences in the design of the OS360 operating system could be considered the birth of "software engineering" as a discipline. Operating systems are also in the origin of "concurrent programming" History of operating systems starts during the sixties with the design of batch systems and later with time sharing systems, like the OS360. The emphasis of operating systems on those days was put on resource sharing, so as to maximize the utilization of the expensive hardware resources. The computer was conceived as a sort of big "electrical factory" supplying computing power to hundreds of terminals. With the appearance and spreading of personal computers, the main interest of operating systems was moved to user interfaces: the introduction of the windowing system and the mouse was considered as a big revolution in the system conception. Finally, networks have supposed the last important milestone in operating systems. The network is actually one of the main resources managed by operating systems. Properly managing it has allowed important facilities like sharing resources across the network in distributed systems. Internet access is nowadays one of the most attractive features of operating systems and web browsers are one of the most important tools.

This course on operating systems mainly covers the main abstractions and the theoretical background of operating systems. For a whole understanding of operating systems it would be highly recommended to complement this course with a second course, more focused on user interfaces, and some course on network programming.

Professor, Dr. Joan Vila

Введение

Важнейшей из целей образовательной системы информационного общества является образование в сфере информационных технологий. Практические достижения в области образовательной информатики, к сожалению, в настоящее время отстают от темпов развития компьютерных технологий, использование которых обусловливается потребностью подготовить учащихся, студентов к их будущей трудовой деятельности, необходимостью более эффективной передачи знаний.

Если принять во внимание быстрое постоянное изменение ситуации на рынке компьютерных технологий, увеличение вдвое количества информации менее чем за полгода, то обучение по более эффективным методикам и умение разработать методику использования современного компьютера в сфере образования становится важным звеном в области образовательной информатики. И только в этом случае возможно сокращение разрыва между появлением нового технического средства, информационной технологии и их эффективного массового использования.

Разработка программ, их реализация, правильное использование всех устройств компьютера – чрезвычайно сложный процесс. Компьютеры оборудованы программным обеспечением, названным *операционной системой*, работа которой должна управлять всеми этими устройствами и обеспечивать пользовательские программы более простым интерфейсом. Операционные системы могут быть характеризованы как интерфейс между пользователями и аппаратными средствами компьютера. В основании интерфейса находятся аппаратные средства, которые во многих случаях сами состоят из двух или более слоев. Самый низкий уровень содержит физические устройства, состоящие из чипов интегральной схемы, проводов, катодных трубок и других подобных физических устройств.

Большинство компьютерных пользователей имеют немного знаний в области операционных систем. Базовые функции операционной системы – это действие операционной системы как расширенной машины и действие операционной системы как менеджера ресурсов. Представляемый курс «Операционные системы» призван помочь пользователям компьютерной приобрести необходимые знания и умения для профессиональной деятельности.

1. Концепция операционной системы

1.1. Компьютерная система

Компьютерная система состоит из процессоров, оперативной памяти, дисков, клавиатуры, монитора, принтеров, сетевого интерфейса и других устройств. Для управления всеми устройствами, корректного и оптимального их использования компьютеры оснащаются специальным уровнем программного обеспечения, которое называется *операционной системой*.

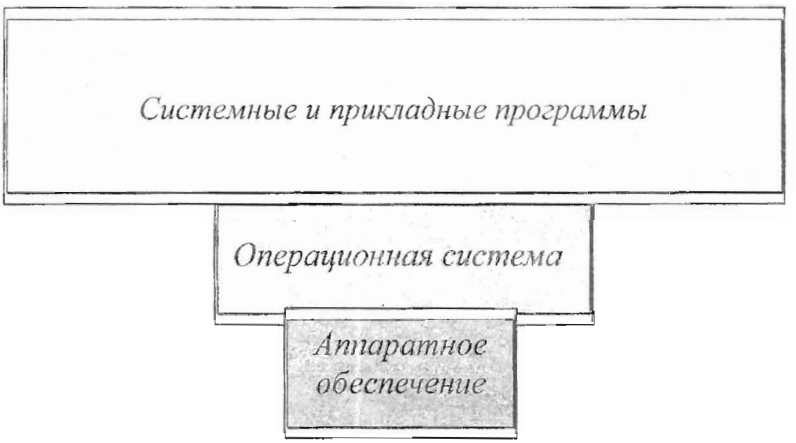


Рис.1.1 Расположение операционной системы в общей структуре компьютера

Операционная система состоит из уровня программного обеспечения, которое частично избавляет от необходимости общения с аппаратурой напрямую. Операционная система отвечает за управление всеми перечисленными устройствами и обеспечивает пользователям простой, доступный интерфейс для работы с аппаратурой. Операционная система управляет и координирует использование аппаратных средств компьютера среди различных прикладных программ для различных пользователей.

Преимущества использования операционных систем:

- удобство
- эффективность

1.2. Операционная система как расширенная машина

Операционная система как расширенная машина. С точки зрения пользователя операционная система выполняет функцию расширенной машины или виртуальной машины, в которой проще программировать и легче работать, чем непосредственно с аппаратным обеспечением, составляющим реальный компьютер. Подведем итог вышесказанному:

операционная система предоставляет нам ряд возможностей, которые могут использовать программы с помощью специальных команд, называемых системными вызовами. Ниже остановимся на вызовах более подробно.

1.3. Операционная система как менеджер ресурсов

Операционная система как менеджер ресурсов. Управление памятью, устройствами ввода-вывода, другими ресурсами и их защита крайне необходимы, когда компьютером (или сетью) пользуются несколько пользователей, поскольку пользователи могут обращаться к ним в абсолютно непредсказуемом порядке. Распределение между пользователями оборудования и информации (файлы, базы данных и т. д.). Основная задача операционной системы заключается в отслеживании того, кто и какой ресурс использует, в обработке запросов на ресурсы, в подсчете коэффициента загрузки и разрешении проблем конфликтующих запросов от различных программ и пользователей. Управление ресурсами включает в себя их распределение во времени и в пространстве.

2. История операционных систем

История развития операционных систем насчитывает большой период времени. Операционные системы появились и развивались в процессе конструирования компьютеров и эти события исторически тесно связаны. Поэтому, чтобы представить, как выглядели операционные системы, рассмотрим следующие друг за другом поколения компьютеров.

Основоположник вычислительной техники английский математик Ч.Бэббидж (*Charles Babbage,* 1792-1871), создавший аналитическую машину. Это была чисто механическая машина, а технологии того времени не были достаточно развиты для изготовления многих деталей и механизмов высокой точности и, конечно, эта аналитическая машина не имела операционной системы.

Первое поколение (1945-55): электронные лампы и коммутационные панели. После попыток Бэббиджа вплоть до Второй мировой войны в конструировании цифровых компьютеров не было практически никакого прогресса. Примерно в середине 1940-х многие ученые в США и в Европе продолжили работу в направлении создания вычислительных машин. На первых машинах использовались механические реле, позже реле заменили электронными лампами. Каждую машину разрабатывала, строила, программировала, эксплуатировала, и поддерживала в рабочем состоянии одна команда. Все программирование выполнялось на абсолютном машинном языке, управления основными функциями машины осуществлялось просто при помощи соединения коммутационных панелей проводами. Тогда еще не были известны языки программирования (даже ассемблера не было). Операционных систем не было.

Второе поколение (1955-65): транзисторы и системы пакетной обработки. В середине 50-х изобретение и применение транзисторов радикально изменило всю картину. Компьютеры стали достаточно надежными, появилась высокая вероятность того, что машина будет работать довольно долго, выполняя при этом полезные функции. Впервые сложилось четкое разделение между проектировщиками, сборщиками, операторами, программистами и обслуживающим персоналом. Машины, теперь называемые *мэйнфреймами*, управляются целым штатом профессиональных операторов, программы на языке *ФОРТРАН* вводились в *ЭВМ* на перфокартах. Общепринятым решением стала система пакетной обработки. Большие компьютеры второго поколения использовались главным образом для научных и технических вычислений, таких, как решение дифференциальных уравнений в частных производных, часто встречающихся в физике и инженерных задачах. Программировали на языке Фортран и ассемблере, а типичными операционными системами были *FMS (Fortran Monitor System)* и *IBSYS* (операционная система, созданная корпорацией *IBM* для компьютера *IBM 7094*).

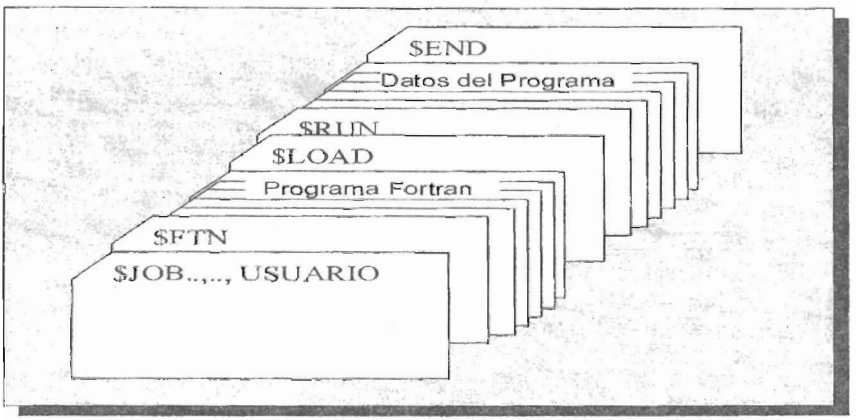


Рис.2.1 Структура типичного задания *FMS*

Третье поколение (1965-1980): интегральные схемы и многозадачность. К началу 60-х годов фирма *IBM* попыталась решить все проблемы разом, выпустив серию программно-совместимых машин *IBM/360* с операционной системой *OS/360*, самым важным достижением которой явилась многозадачность, другим важным плюсом операционных систем третьего поколения стала их способность загружать новое задание с диска в освободившийся раздел памяти и запускать его. Этот технический прием называется «подкачкой» данных или spooling (*Simultaneous Peripheral Operation On Line* – совместная периферийная операция в интерактивном режиме). Для сокращения времени ожидания ответа начата разработка систем с режимом разделения времени, первая из них *CTSS* (*Compatible*

Time Sharing System – Совместимая система разделения времени). Операционная система, которая должна была поддерживать сотни одновременных пользователей в режиме разделения времени известна как *MULTICS* (*MULTiplexed Information and Computing Service* – мультиплексная информационная и вычислительная служба), значительно повлиявшая на последующие операционные системы. Усеченная однопользовательская версия системы *MULTICS* позже развилась в операционную систему *UNIX*®, ставшую популярной в академическом мире, в правительственных управлениях и во многих компаниях. Чтобы стало возможным писать программы, работающие в любой *UNIX*-системе, Институт инженеров по электротехнике и электронике *IEEE* разработал стандарт системы *UNIX*, называемый *POSIX*, который теперь поддерживают большинство версий *UNIX*. Стандарт *POSIX* определяет минимальный интерфейс системного вызова, который должны поддерживать совместимые системы *UNIX*. Некоторые другие операционные системы теперь тоже поддерживают интерфейс *POSIX*.

В 1987 году создан маленький клон системы *UNIX* для образовательных целей, так называемая система *MINIX*, на основе которой разработана система *Linux*.

Четвертое поколение (с 1980 года по наши дни): персональные компьютеры. Следующий период в эволюции операционных систем связан с появлением Больших Интегральных Схем (*LSI, Large Scale Integration*) – кремниевых микросхем, содержащих тысячи транзисторов на одном квадратном сантиметре.

В 1974 году, когда компания *Intel* выпустила *Intel 8080*, первый универсальный 8-разрядный центральный процессор, для него потребовалась операционная система, с помощью которой можно было бы протестировать новинку, ею была дисковая операционная система, названную *CP/M* (*Control Program for Microcomputers* – программа управления для микрокомпьютеров). В начале 80-х корпорация *IBM* разработала *IBM PC* (*Personal Computer* – персональный компьютер), подходящей операционной системой стала система *DOS* (*Disk Operating System* – дисковая операционная система), впоследствии *MS-DOS* (*MicroSoft Disk Operating System*). *CP/M, MS-DOS* и другие операционные системы для первых микрокомпьютеров полностью основывались на вводе команд с клавиатуры. Затем это свойство операционных систем изменилось – изобретен графический интерфейс пользователя (*GUI, Graphical User Interface*), состоящий из окон, значков, различных меню и мыши, создан *Apple* с графическим интерфейсом. Корпорация *Microsoft* создала преемника *MS-DOS* под влиянием успехов компании *Macintosh*, разработана система, получившая название *Windows*, базой для которой послужил *GUI*. Версии этой системы – *Windows 95, Windows 98 , Windows NT* (*NT* означает *New Technology* – новая технология) *Windows Me* (*Millennium edition* – выпуск тысячелетия).

Главным соперником *Windows* в мире персональных компьютеров становится система *UNIX* и ее различные производные. *UNIX* является самой сильной системой для рабочих станций и других компьютеров старших моделей, таких, как сетевые серверы. Она особенно популярна на машинах с высокопроизводительными *RISC*-процессорами (*RISC, reduced instruction set computer* – компьютер с сокращенным набором команд). На компьютерах с процессорами *Pentium* популярной альтернативой *Windows* для студентов и других разнообразных пользователей становится *Linux*.

С середины 80-х годов начали расти и развиваться сети персональных компьютеров, управляемых сетевыми и распределенными операционными системами. Каждый компьютер работает под управлением локальной операционной системы и имеет своего собственного локального пользователя (или пользователей). Сетевые операционные системы несущественно отличаются от однопроцессорных операционных систем.

Распределенная операционная система, напротив, представляется пользователям традиционной однопроцессорной системой, хотя она и составлена из множества процессоров. Распределенные системы, например, часто позволяют прикладным задачам одновременно обрабатываться на нескольких процессорах, поэтому требуется более сложный алгоритм загрузки процессоров для оптимизации распараллеливания.

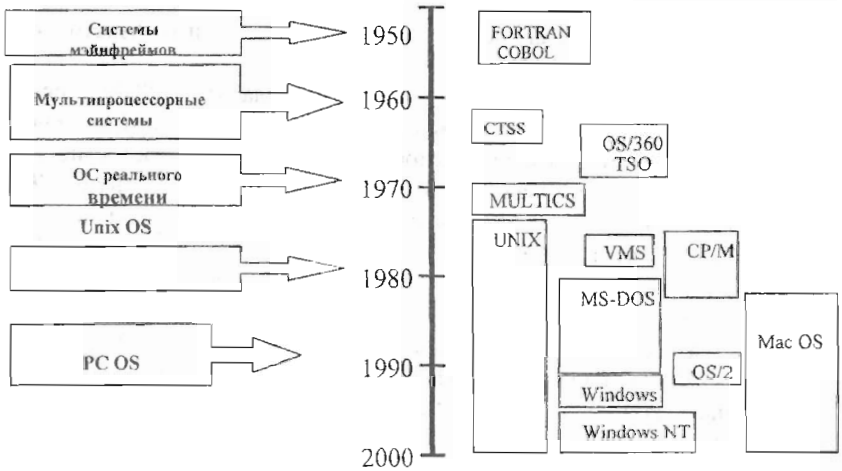


Рис.2.2 Эволюция операционных систем

2.1. Операционные системы мэйнфреймов; простые системы

Основные особенности:

- отсутствие прямого взаимодействия пользователь – компьютер;
- объединение большого количества одинаковых действий;

- автоматическая последовательность работ управления;
- функции операционной системы.

Резидентский монитор:

- ссылки прерываний;
- управление устройствами;
- последовательность работы;
- переводчик языка управления.

Проблемы простых систем возникают из-за больших значений времени ввода-вывода из-за низкого использования центрального процессора.

Способы решения:

- использование подкачки
- мультипрограммирование

2.2. Серверные операционные системы

Уровнем ниже находятся серверные операционные системы. Они работают на серверах, одновременно обслуживают множество пользователей и позволяют им делить между собой программные и аппаратные ресурсы. Серверы предоставляют возможность работы с печатающими устройствами, файлами или Интернетом. Интернет-провайдеры обычно запускают в работу несколько серверов для того, чтобы поддерживать одновременный доступ к сети множества клиентов. На серверах хранятся страницы web-сайтов и обрабатываются входящие запросы. *UNIX* и *Windows* 2000 являются типичными серверными операционными системами. Теперь в этих целях стала использоваться и операционная система *Linux*.

2.3. Многопроцессорные операционные системы

Особенности *многопроцессорных* операционных систем:

- Распределение основной памяти
- Поочередное использование центрального процессора

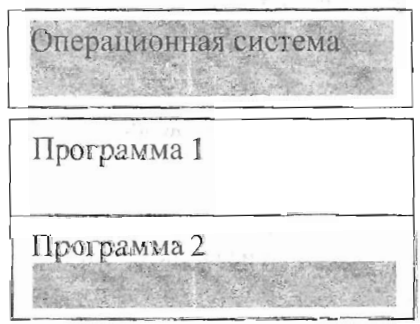


Рис.2.3 Структура многопроцессорной операционной системы

Операционная система обладает большими размерами, появляются новые функции:

- планирование работ;
- планирование управления памяти центрального процессора.

2.4. Операционные системы реального времени

Основные функции операционных систем:

- управление процессами;
- управление памяти;
- управление файлами;
- защита.

Современные тенденции разработки систем реального времени:

- ограничения во времени;
- критические системы в режиме реального времени.

2.5. Операционные системы персональных компьютеров

Основное назначение – максимизация эксплуатационной надежности пользователя и быстродействия:

- типичные системы (*Windows, Linux, MacOS*);
- интерфейс базируется на системе окон и мыши;
- системы файлов в виде дерева (*Unix*);
- возможность мультизадач;
- мультимедийные возможности
- доступ к сети;
- механизмы защиты.

Современные тенденции разработки операционных систем персональных компьютеров:

1. параллельные системы;
2. мультипроцессорные системы.

преимущества:

- увеличение быстродействия и объема вычислений;
- надежность.

3. распределенные системы для распределения вычислений между несколькими физическими процессорами

преимущества:

- распределение ресурсов;
- увеличение быстродействия вычислений;
- надежность;
- связь.

4. сетевые операционные системы

3. Основные функции операционной системы

Операционная система тесно связана с оборудованием компьютера. Аппаратное обеспечение влияет на набор команд операционной системы и управление его ресурсами.

3.1. Обзор аппаратного обеспечения компьютера

Механизмы прерывания операционной системы. Действия менеджера прерывания:

- экономия направления инструкции;
- дисквалификация достижений новых прерываний;
- другие действия.

Возможные проблемы возникают из-за прерываний, вызванных ошибкой или запросом пользователя.

Операции в двух направлениях.

Направления работы центрального процессора:

- направление пользователь: ограниченная подкомиссия;
- направление супервизор (ядро, монитор или система).

Рис. 3.1. Схема операций в двух направлениях

Привилегированные инструкции:

- инструкции связаны преимущественно с тремя типами защиты:

A) защита Входа / Выход;
B) защита памяти;
C) защита процессора.

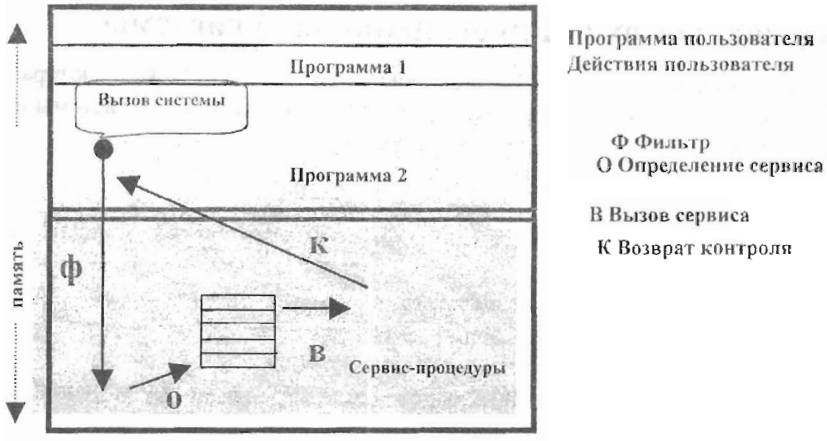


Рис. 3.2 Архитектура аппаратных средств

3.2. Сервис операционной системы

1. Операционная система как расширенная машина:
- выполнение программ;
- действия ввода – вывода;
- использование системы файлов;
- связь;
- обнаружение ошибок.
2. Операционная система как менеджер ресурсов:
- учет;
- защита.

В качестве примера рассмотрим систему *Unix*.

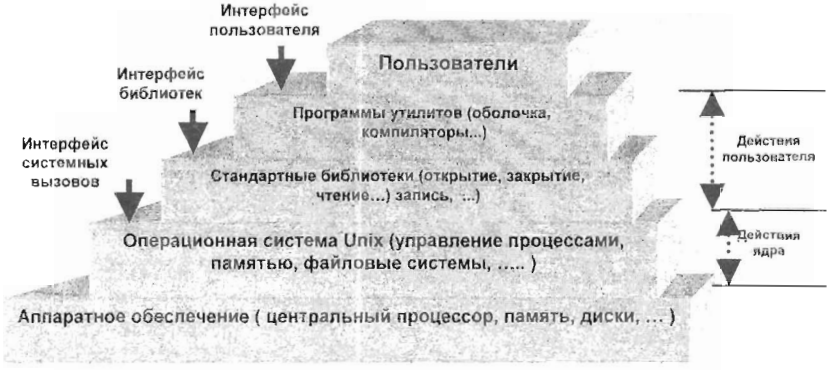


Рис. 3.3 Интерфейсы в *UNIX*

3.3. Системные вызовы

Набор системных вызовов определяет интерфейс между операционной системой и программами пользователя. Системные вызовы операционных систем отличаются, однако, в основе их функционирования – одна концепция.

Различные типы системных вызовов:

- системные вызовы для управления процессами;
- системные вызовы для управления файлами;
- системные вызовы для управления устройствами;
- информационное обслуживание;
- связь.

3.3.1. Системные вызовы для управления процессами

Основные выполняемые функции:

- завершение, аварийное прекращение работы;
- загрузка, выполнение;
- создание процесса, завершение процесса;
- получение атрибутов процесса, установление атрибутов процесса;
- ожидание в течение времени;
- ожидание сигнала;
- размещение и освобождение памяти.

3.3.2. Системные вызовы для управления файлами

Основные выполняемые функции:

- создание файла, удаление файла;
- открытие, закрытие;
- чтение, запись, перемещение;
- получение атрибутов файла, установление признаков файла.

3.3.3. Системные вызовы для управления устройствами

Основные выполняемые функции:

- запрос устройства, выпуск устройства;
- чтение, запись, перемещение;
- получение атрибутов устройства, установление признаков устройства;
- логическое присоединение или разъединение устройства.

3.3.4.Информационное обслуживание

Основные выполняемые функции:

- получение и установление времени и даты;
- получение и установление данных системы;
- получение процесса, файла или признаков устройств;
- процессы набора, файла или признаков устройств.

3.3.5. Связь

Основные выполняемые функции:

- создание, удаление связи;
- посылка и получение сообщения;
- информация статуса передачи;
- приложение или отделение отдаленных устройств.

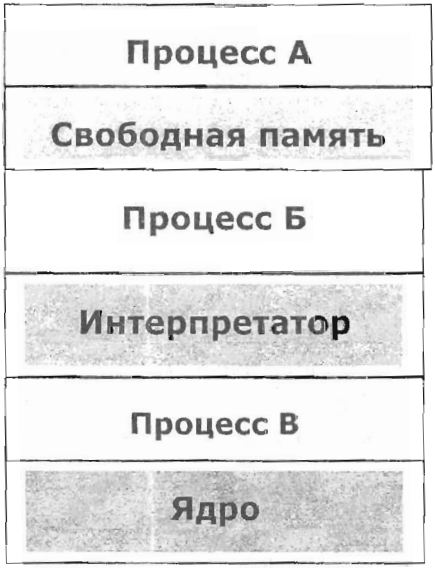


Рис. 3.4 *UNIX* – многократные программы

3.4. Системные программы

Концепция системных программ. Системные программы служат для обслуживания операционной системы, выполнены как процессы пользователя и обеспечивают более удобную окружающую среду. Представляют собой программы, написанные на языке программирования (например, язык С). Функции системных программ:

- манипуляция файлами;
- информация о состоянии;
- модификация файлов;
- использование языков программирования;
- загрузка и выполнение программ;
- связь.

3.4.1. Интерпретатор команд или оболочка *(shell)*

Концепция интерпретатора команд. Интерпретатор команд, первичный интерфейс между пользователем и операционной системой, представляет собой программу, которая читает команду (или файл текста с многократными командами), представленный пользователем, анализирует и выполняет команду. Интерпретатор команд или оболочка читает команды с терминала

При входе пользователя в систему, запускается оболочка. Стандартным входным и выходным устройством для оболочки является терминал (монитор с клавиатурой). Рассмотрим командный интерпретатор *UNIX,* называемый оболочкой *(shell)* и являющийся хорошим примером того, как могут применяться системные вызовы. Кроме этого, оболочка предоставляет основной интерфейс между пользователем, сидящим за своим терминалом, и операционной системой, если пользователь не использует графический интерфейс. Существует множество оболочек, например *sh, csh, ksh и bash.* Все они поддерживают описанные ниже функции, поскольку произошли от первоначальной оболочки *(sh).*

Командный интерпретатор *UNIX:*

- Программирование оболочки *UNIX;*
- *UNIX,* переводчик заказов – программа, которая выполнена как процесс пользователя;
- примеры: *sh, ksh, csh, …*

Характеристики интерпретатора команд:

- внутренние и внешние заказы;
- выполнение;
- управление потоками;
- сценарии оболочки.

3.5. Структура операционной системы

Рассмотрим операционную систему изнутри, пять различных структур операционной системы:

- монолитная система;
- многоуровневая система;
- виртуальные машины;
- экзоядро;
- модель клиент-сервер.

3.5.1. Монолитная система

Монолитная система написана в виде набора процедур, каждая из которых имеет строго определенный интерфейс и возможность вызвать любую другую процедуру. Простая модель монолитной системы: утилиты, сервисные процедуры, главная процедура.

3.5.2. Многоуровневые системы

Многоуровневые системы организованы в виде иерархии уровней. В качестве примера система *THE* (автор Э.Дейкстра), состоящая из 6 уровней: от распределения процессора и многозадачности до процесса.

3.5.3. Виртуальные машины

Основная часть системы, называемая монитором виртуальной машины, работает с оборудованием и обеспечивает многозадачность, предоставляя не одну, а несколько виртуальных машин.

Рис. 3.5 Модели систем. а) реальная машина, б) виртуальная машина

3.5.4. Экзоядро

Операционная система обеспечивает каждого пользователя копией реального компьютера, но с множеством ресурсов. На нижнем уровне, на уровне ядра работает программа экзоядро, предоставляющая распределение ресурсов для виртуальных машин, проверка их использования. Преимущество схемы экзоядра в том, что она функционирует без отображения.

3.5.5. Модель клиент-сервер

В современных операционных системах тенденция в сторону переноса кода в верхние уровни, оставляя минимальное микроядро. Получая запрос на операцию пользовательский процесс (клиентский запрос) посылает запрос серверному запросу, который его обрабатывает и высылает назад ответ.

4. Процессы

Процесс -- ключевое понятие операционной системы. Исполняемая программа, включая текущие значения счетчика команд, регистров и переменных представляет собой процесс. С каждым процессом связано понятие его адресного пространства, которое включает в себя саму программу, данные и стек программы.

4.1. Концепция процесса

В современных компьютерах могут выполняться одновременно несколько действий: чтение с диска, вывод текста на монитор или принтер. При этом в каждый момент времени один процессор занят только одной программой, но за секунду выполняется несколько программ. Для управления одновременно выполняемыми процессами в операционных системах разработана концептуальная модель последовательных процессов.

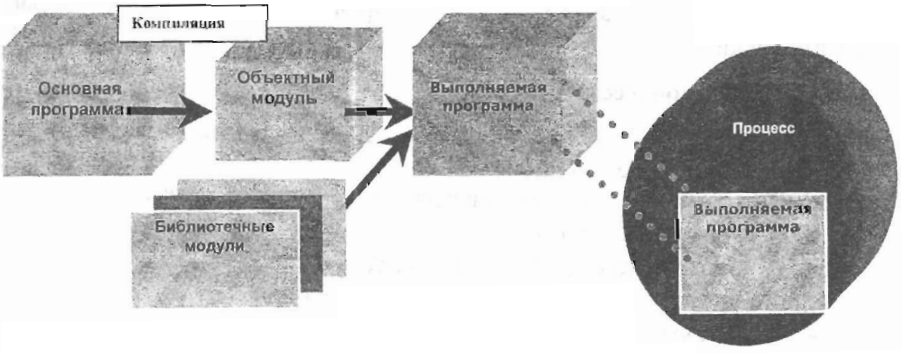


Рис. 4.1. Реализация программы

4.2. Модель процесса

Программное обеспечение современных компьютеров, в отдельных случаях включая операционную систему, организовано в виде набора последовательных процессов, представляющих собой выполняемые программы. Реальный процессор переключается с процесса на процесс, это переключение носит название *многозадачность или мультипрограммирование.*

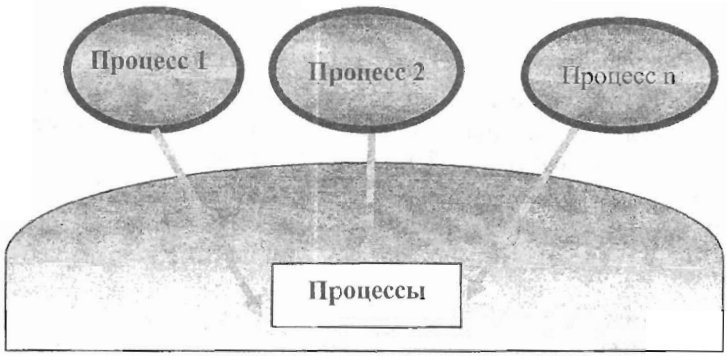


Рис. 4.2. Последовательное выполнение процессов

В понятие модели процесса входит:

1. Создание процессов системы.
2. Планирование процессов.
3. Условия механизмов для синхронизации, связи, обработки взаимоблокировок.

4.3. Создание процесса

В операционной системе необходим способ создания и прерывания процессов по мере необходимости.

Четыре основных события, приводящие к созданию процесса:

- инициализация системы;
- выполнение запроса системы на создание процесса;
- пользовательский запрос на создание нового процесса;
- инициирование пакетного задания;

4.4. Завершение процесса

Новый процесс завершится благодаря одному из следующих событий:

- нормальный выход (преднамеренно);
- выход в результате ошибки (преднамеренно);
- выход в результате неисправимой ошибки (непреднамеренно);
- уничтожение другим процессом (непреднамеренно).

Состояния процесса.

Пример программы

Команда *Shell*

"Cat chapter1 chapter2 chapter3 | grep tree"

- первый процесс *cat* (соединяет и выводит три файла)
- второй *grep* выбирает все линии, содержащие слово *«tree»*.

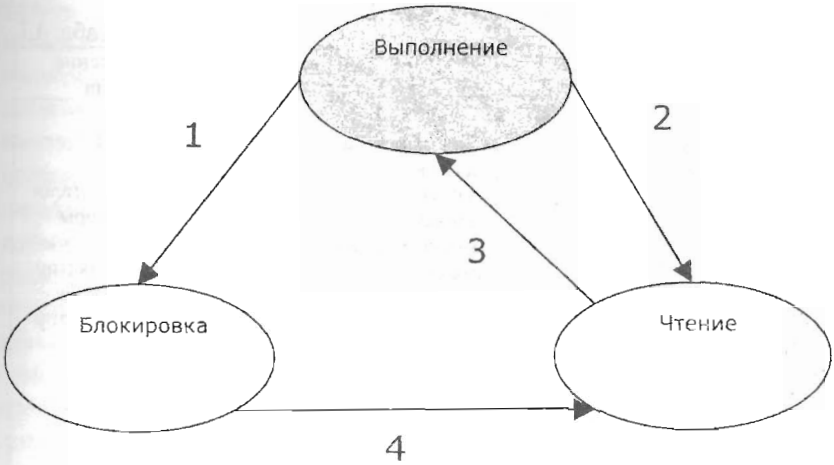


Рис.4.3 Состояния процесса: 1 – процесс блокируется для входа; 2 – планировщик находит другой процесс; 3 – планировщик выбирает этот процесс; 4 – вход становится доступным.

Характеристику различных состояний процессов можно перечислить:

- новый – процесс создается;
- выполнение – инструкции выполняются;
- ожидание – процесс ждет некоторого случая, чтобы произойти;
- готовность – процесс ждет, чтобы быть назначенным на процессор;
- завершение – процесс завершился выполненным.

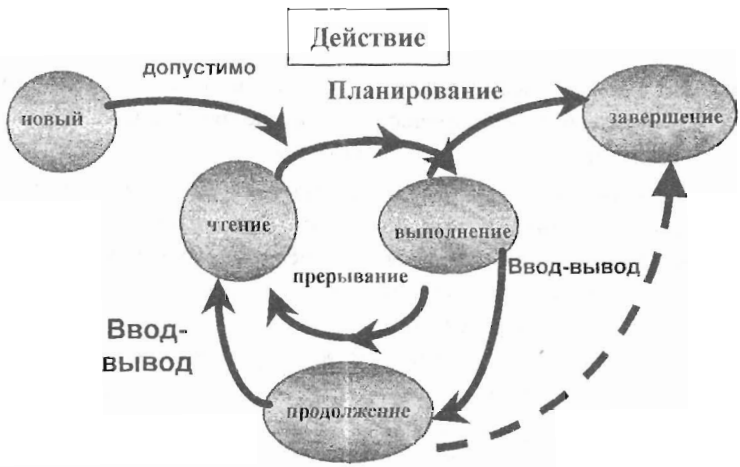


Рис. 4.4 Диаграмма состояний процесса

4.5. Реализация процессов

Типичная таблица процессов

Табл. 4.1

Управление процессом	Управление памяти	Управление файла
• Регистры • Счетчик команд • указатель стека • состояние процесса • приоритет • планирование параметров • обработка ID • родительский процесс • группа процесса • сигналы • время начала процесса • использованное время центрального процессора • процессорное время дочернего процесса • время следующего аварийного сигнала	Указатель на текст овый сегмент указатель на сегмент данных Указатель на сегмент стека	Корневой каталог Рабочий каталог Дескрипторы файла Идентификатор пользователя Идентификатор группы

5. Потоки

Обычно процессу соответствует адресное пространство и одиночный управляющий поток, фактически это определяет процесс. Однако, возникают ситуации, когда предпочтительно иметь несколько квазипараллельных управляющих потоков в одном адресном пространстве.

5.1. Модель потока

Процесс можно рассматривать, как поток исполняемых команд. Различные потоки в одном процессе не так независимы, как различные процессы. Ниже представлена модель потоков.

Модель процесса, которую мы рассматривали, базируется на двух независимых концепциях: группировании ресурсов и выполнении программы. Иногда полезно разделять и здесь возникает понятие потока.

У процесса есть адресное пространство, содержащее текст программы и данные, а также другие ресурсы.

С другой стороны процесс можно рассматривать как поток исполняемых команд или просто поток, содержащий счетчик команд, регистры. Поток содержит счетчик команд, отслеживающий порядок исполнения действий, регистры, в которых хранятся текущие переменные, стек, содержащий протокол выполнения процесса, где на каждую процедуру, вызванную, но еще не вернувшуюся, отведен отдельный

фрейм. Поток должен исполняться внутри процесса. Процессы используются для группирования ресурсов, а потоки являются объектами, поочередно исполняющимися на центральном процессоре. Концепция потоков добавляет к модели процесса возможность одновременного выполнения в одной и той же среде процесса нескольких программ, в достаточной степени независимых. Несколько потоков, работающих параллельно в одном процессе, аналогичны нескольким процессам, идущим параллельно на одном компьютере. В первом случае потоки разделяют адресное пространство, открытые файлы и другие ресурсы. Во втором случае процессы совместно пользуются физической памятью, дисками, принтерами и другими ресурсами. Потоки обладают некоторыми свойствами процессов, поэтому их иногда называют упрощенными процессами. Термин *многопоточность* также используется для описания использования нескольких потоков в одном процессе.

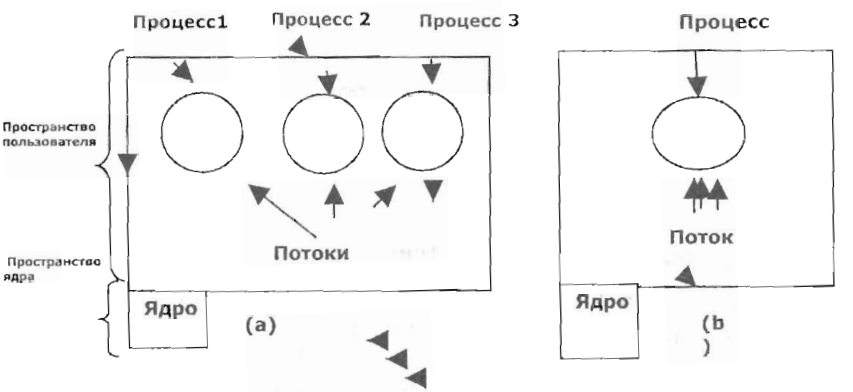


Рис. 5.1. (а) Три процесса с одиночными потоками управления. (б) Один процесс с тремя потоками управления

Табл. 5.1

Элементы процесса, совместно используемые всеми потоками процесса	Элементы потока, индивидуальные для каждого потока
Адресное пространство	Счетчик команд
Глобальные переменные	Регистры
Открытые файлы	Стек
Дочерние процессы	Состояние
Необработанные аварийные сигналы	
Сигналы и их обработчики	
Информация об использовании ресурсов	

5.2. Многопоточная модель

В многопоточном режиме процессы, как правило, запускаются с одним потоком. Этот поток может создавать новые потоки.

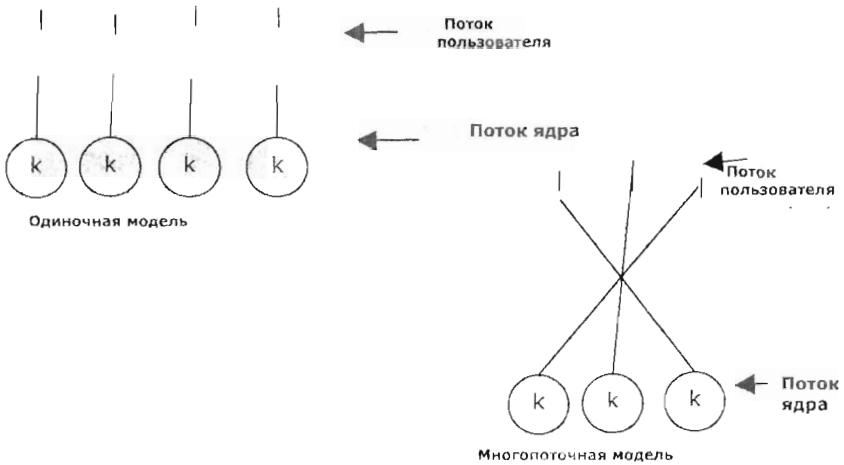


Рис.5.2 Многопоточная модель.

5.3. Межпроцессорное взаимодействие

Процессы часто взаимодействуют между собой и здесь необходимо правильно организовать взаимодействие между процессами, по возможности не используя прерывания. Процесс может создавать несколько других процессов (называющихся дочерними процессами), а эти процессы, в свою очередь, тоже могут создать дочерние процессы, то есть предстает дерево процессов. Связанные процессы – это те, которые объединены для выполнения некоторой задачи, им нужно часто передавать данные от одного к другому и синхронизировать свою деятельность. Такая связь и называется *межпроцессорным взаимодействием.*

5.3.1. Состояние состязания

В некоторых операционных системах процессы, работающие совместно, могут использовать общее хранилище данных. Ситуации, в которых несколько процессов считывают или записывают данные одновременно, называются *состояниями состязания.*

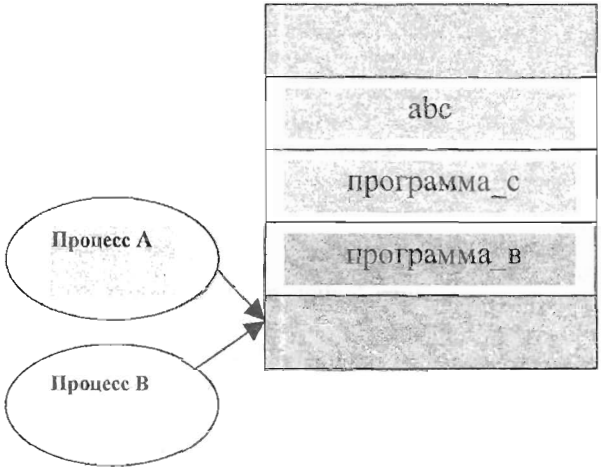


Рис 5.4. Два процесса хотят одновременно получить доступ к совместно используемой памяти.

5.3.2. Семафоры и мьютексы

Примитив синхронизации, семафор – новый тип переменных, значение которых может быть нулем (в случае отсутствия сохраненных сигналов активизации) или некоторым положительным числом, соответствующих количеству отложенных активизирующих сигналов. Две операции *dow* и *up* уменьшают и увеличивают значения семафора.

Мьютекс – упрощенная версия семафора, переменная, которая может находиться в блокированном или неблокированном состоянии и управляет взаимным исключением доступа к совместно используемым ресурсам.

5.3.3. Мониторы

Монитор – набор процедур, переменных и других структур, обединенных в особый модуль или пакет.

monitor *example* **integer** i;
condition *c;*
procedure *producer^);*
end;
procedure *consumer^*);
end;

- end monitor:

5.3.4. Барьеры

Механизмы синхронизации, предназначенные для групп процессов, в ситуациях, когда процесс не может перейти в следующую фазу, пока к этому не готовы все остальные процессы. Этого можно добиться, разместив в конце каждой фазы барьер.

5.4. Планирование

При многозадачной работе компьютера, в нем могут быть несколько процессов, пытающихся получить одновременный доступ к процессору. Если доступен один процессор, необходимо выбирать между процессорами, это и является основной функцией планировщика, а используемые алгоритмы – алгоритмы планирования. При этом важно обращать внимание на:

- поведение процесса, когда необходимо планировать
- категории алгоритмов планирования
- задачи алгоритмов планирования
- планирование в системах пакетной обработки данных
- планирование в интерактивных системах.

6. Взаимоблокировка

При выполнении прикладных задач процесс нуждается в доступе не к одному, а к нескольким ресурсам. Когда взаимодействуют два или более процессов, они могут попадать в патовые ситуации, из которых невозможно выйти без посторонней помощи. Такая ситуация называется тупиком, тупиковой ситуацией или *взаимоблокировкой*.

Компьютерные процессы могут попадать в аналогичные ситуации, в которых они не могут продвигаться дальше. Представьте себе компьютер с накопителем на магнитной ленте и записывающим компакт-диски устройством *(CD-recorder)*. Теперь представьте, что каждому из двух процессов нужно записать данные с ленты на компакт-диск. Процесс 1 запрашивает и получает в пользование устройство с лентой. Затем процесс 2 запрашивает и получает устройство для записи компакт-дисков. После этого процесс 1 запрашивает устройство для записи компакт-дисков и приостанавливается до тех пор, пока процесс 2 не освободит его. Наконец, процесс 2 запрашивает устройство с лентой и также останавливается на время, потому что магнитофон уже занят процессом 1. Перед нами типичный тупик, из которого нет выхода.

6.1. Условия взаимоблокировки

Чтобы произошла взаимоблокировка, должны выполниться следующие условия:

- Условие взаимного исключения. Каждый ресурс в данный момент или отдан только одному процессу или доступен.
- Условие удержания и ожидания. Процессы, удерживающие полученные ранее ресурсы, могут запрашивать новые ресурсы.
- Условие отсутствия принудительной выгрузки ресурса. Нельзя принудительным образом у процесса забрать полученные ранее ресурсы. Процесс, владеющий ими, должен сам освободить ресурсы.
- Условие циклического ожидания. Должна существовать круговая последовательность из двух или более процессов, каждый из которых ждет доступа к ресурсу.

6.2. Страусовый алгоритм

Является самым простым подходом. Суть алгоритма в том, что большая часть операционных систем, в том числе *Windows* и *Unix* игнорирует проблему взаимоблокировок.

6.3. Обнаружение и устранение взаимоблокировок

Система не пытается предотвратить попадание в тупиковые ситуации. Система предпринимает меры к возвращению системы к состоянию до попаданию в тупик.

6.4. Избежание взаимоблокировок

- Траектории ресурса.
- Безопасные и небезопасные состояния.
- Алгоритм банкира для одного вида ресурса.
- Алгоритм Банкира для нескольких видов ресурсов.

6.5. Предотвращение взаимоблокировок

- Атака условия взаимного исключения.
- Атака условия удержания и ожидания.
- Атака условия отсутствия принудительной выгрузки ресурса.
- Атака условия циклического ожидания.

6.6. Сопутствующие вопросы

- Двухфазовое блокирование.
- Тупики без ресурсов.
- Голодание.

7. Управление памятью

Компьютеры имеет иерархию памяти, включающей в себя очень быструю кэш-память, десятки мегабайтов средней скорости оперативную

память, десятки или сотни гигабайтов медленной памяти на диске. Работа операционной системы заключается в координировании, как эти блоки памяти используются. Часть операционной системы, которая управляет иерархией памяти, называют менеджером памяти. Функции менеджера памяти заключаются в определении, какие части памяти используются и какие части нет, распределении памяти между процессами, когда они нуждаются в этом. Менеджер памяти управляет подкачкой между оперативной памятью и диском, когда оперативная память является слишком маленькой, чтобы управлять всеми процессами.

7.1. Основное управление памятью

- однозадачная система без подкачки на диск;
- многозадачность с фиксированными разделами;
- моделирование многозадачности;
- настройка адресов и защита;
- подкачка.

Самая простая возможная схема управления памятью состоит в том, чтобы выполнить только одну программу одновременно, совместно используя память между программой и операционной системой. Три разновидности этой темы показано на рисунке:

Рис.7.1 Однозадачная система без подкачки на диск.

7.2. Подкачка

Два общих подхода к управлению памятью могут использоваться, в зависимости от доступных аппаратных средств. Самая простая стратегия, названная *подкачкой*, состоит из введения каждого процесса полностью, выполнение этого некоторое время и размещение на диске.

Операция подкачки системы иллюстрирована в *рис.7.2.*, где представлена конфигурация памяти, в которой пространство для роста было распределено между двумя процессами.

На рисунке мы видим, что каждый иллюстрированный процесс имеет стек наверху его распределенной памяти и сегмент данных вне текста программы. Память между ними может использоваться для любой области.

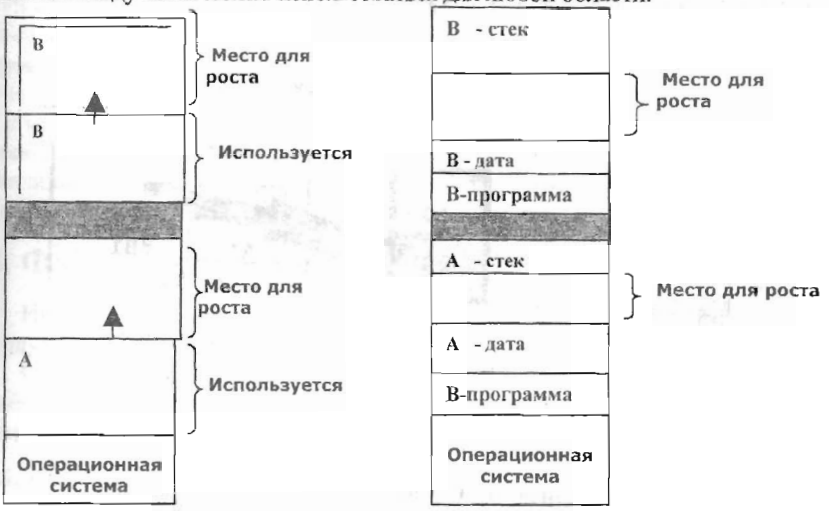


Рис 7.2 (а) Предоставление пространства для роста области данных;
(б) Предоставление пространства для роста стека области данных

7.2.1. Управление памятью битовыми массивами

Если память выделена динамически, операционная система должна управлять этим процессом. Существует два способа учета использования памяти: битовые массивы и списки свободных участков. При работе с битовыми массивами память разделена на модули от нескольких слов до нескольких килобайтов. В битовом массиве каждому свободному блоку соответствует один бит, равный нулю, а каждому занятому бит, равный 1.

Битовый массив предоставляет простой способ отслеживания слов в памяти фиксированного обьема.

7.2.2.Управление памятью с помощью связных списков

Другой способ следить за памятью состоит в том, чтобы обслужить связные списки распределенных и свободных фрагментов памяти, где фрагментом является или процесс или участок между двумя процессами.

Пусть список отсортирован по адресам, сортировка этого пути имеет преимущество, что, когда процесс заканчивается или скачивается на диск, изменение списка представляет собой несложную операцию.

7.3. Виртуальная память

Основная идея виртуальной памяти – объединенный размер программы, данных, и стека может превысить количество физической памяти. Операционная

система сохраняет те части программы, используемой в настоящее время в оперативной памяти, остальные части находятся на диске.

Виртуальная память может также работать в многозадачной системе.

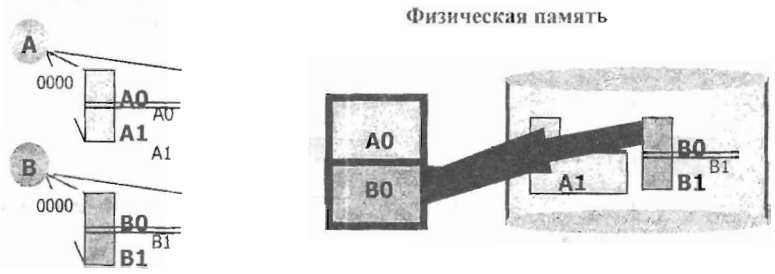


Рис 7.3 Представление виртуальной и физической памяти.

7.3.1. Страничная организация памяти

Рис 7.4. Страничная организация памяти

7.4. Алгоритмы замещения страниц

Когда происходит страничное прерывание, операционная система должна выбрать страницу для удаления из памяти, чтобы создать место для страницы, которая должна быть введена. Возможные алгоритмы замещения страниц:

- оптимальный алгоритм;
- алгоритм «не использованная в последнее время страница;
- «первым прибыл – первым обслужен»;
- алгоритм «вторая попытка»;
- алгоритм «часы»;
- алгоритм «дольше всего не используемая страница»;
- алгоритм *«WSClock»*;

7.5. Сегментация

Сегментированная память обладает простотой управления, увеличивающимися или сокращающимися структурами данных. Решение состоит в том, чтобы обеспечить машину множеством полностью независимых адресных пространств – сегментами. Сегмент – логический объект, который может иметь в составе процедуру, массив, стек или набор скалярных переменных. Сегментация облегчает совместное использование процедур и данных несколькими процессами. Общий пример – библиотека совместного доступа. Реализация сегментации существенно отличается от страничной организации памяти: страницы имеют фиксированный размер, а сегменты – нет.

8. Файловые системы

Информация хранится на дисках и других внешних носителях в модулях, называемых *файлами*.

Файловой системой называется часть операционной системы, работающая с файлами. Важным аспектом файловой системы является ее внешнее представление, операции с файлами, защита файлов и др.

8.1. Файлы.

Файлы представляют способ сохранять информацию на диске и считывать ее. Важной характеристикой является именование файлов. При создании файла процесс дает файлу имя, которое сохраняется и после завершения процесса. К файлу по его имени могут получить доступ другие процессы. Все современные операционные системы поддерживают использование в качестве имен 8-символьные текстовые строки. Многие файловые системы поддерживают имена файлов до 255 символов. В *UNIX* различаются прописные и строчные символы, во всех версиях *Windows* и *MS-DOS* нет. Во многих операционных системах имя файла состоит из двух частей, разделенных точкой.

Часть имени файла после точки называется расширением и обычно означает тип файла. В системе *UNIX* размер расширения зависит от пользователя, а система *Windows*, напротив, назначает каждому расширению определенное значение.

8.1.1. Структура файла

Файлы могут быть структурированы различными способами: файл представляет собой последовательность байтов, последовательность записей, дерево. В первом случае операционную систему не интересует содержимое файла, она видит только байты. Значения байтам придаются программами уровня пользователя (в системах *UNIX* и *Windows*). Рассмотрение операционной системой файлов просто как последовательности байтов обеспечивает максимальную гибкость.

В структуре файла как последовательности записей важным является то, что операция чтения возвращает одну запись, а операция записи перезаписывает или дополняет одну запись.

Третий вариант структуры данных в виде дерева записей, не обязательно одной и той же длины. Каждая запись содержит поле ключа, и дерево отсортировано по ключевому полю, что обеспечивает быстрый поиск заданного ключа. Основная файловая операция здесь не получение следующей записи, а получение записи с указанным значением ключа.

В современных операционных системах все файлы автоматически являются файлами произвольного доступа.

8.1.2. Атрибуты файла

- Защита.
- Пароль.
- Создатель.
- Владелец.
- Флаг «только чтение».
- Флаг «скрытый».
- Флаг»системный».
- Флаг «архивный».
- Флаг ASCII/ двоичный.
- Флаг произвольного доступа.
- Флаг временный.
- Флаг блокировки.
- Длина записи.
- Позиция ключа.
- Длина ключа.
- Время создания.
- Время последнего доступа.
- Время последнего изменения.
- Текущий размер.
- Максимальный размер.

8.1.3. Операции с файлами

1. Создать. Файл создан без данных. Системный запрос объявляет о появлении нового файла и позволяет установить некоторые из его признаков.

2. Удалить. Когда файл больше не необходим, это должно быть удалено, чтобы освободить место диска.

3. Открыть. Перед использованием файла, процесс должен открыть это.

4. Закрыть. Все доступы закончены, признаки и диск адреса больше не необходимы, так что файл должен быть закрыт, чтобы освободить место.

5. Читать. Данные читаются от файла. Вызывающий должен определить, насколько данные необходимы и должен также обеспечить буфер, чтобы вставить их.

6. Записать. Запись данных в файл.

7. Добавить Усеченная форма вызова запись, может только добавить данные к концу файла.

8. Искать. Для случайных файлов доступа, метод необходим, чтобы определить, где взять данные.

9. Получение атрибутов. Процессам часто необходимо получить атрибуты файла.

10. Установа атрибутов. Некоторые из признаков – могут быть изменены после того, как файл был создан.

11. Переименовать. Возникает необходимость, когда пользователь должен изменить название существующего файла.

8.1.4. Реализация файлов

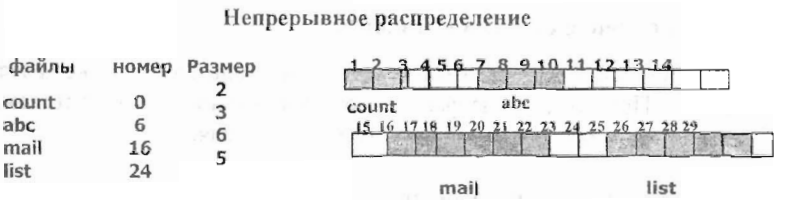


Рис. 8.1 Схема расположения непрерывных файлов на диске

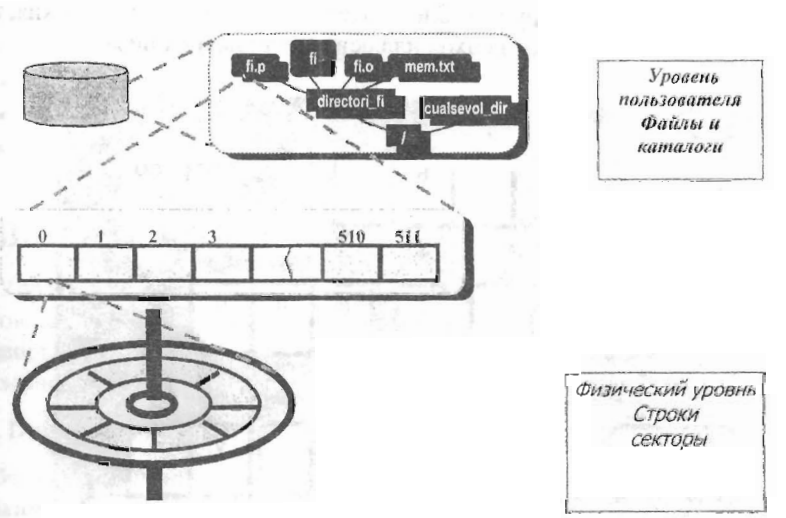


Рис.8.2. Схема выполнения файлов

8.2. Каталоги

В файловых системах файлы обычно организуются в каталоги.

8.2.2. Одноуровневые каталоговые системы.

Простейшая форма системы каталогов, когда имеется один корневой каталог, в котором содержатся все файлы.

Недостатком системы с одним каталогом в том, что различные пользователи могут случайно использовать для своих файлов одинаковые имена.

8.2.3. Двухуровневая система каталогов

Рассмотрим систему, в которой каждому пользователю выделяется один каталог. При этом имена файлов, созданных одним пользователем, не конфликтуют с именами файлов другого пользователя.

8.2.4. Иерархические каталоговые системы

Благодаря двухуровневой иерархии исчезают конфликты имен файлов между различными пользователями, но она недостаточна для пользователей с большим числом файлов, когда возникает потребность логически группировать файлы. И здесь востребован гибкий способ, позволяющий объединять эти файлы в группы. Следовательно, нужна общая иерархия, то есть дерево каталогов. Этот подход иллюстрирует рисунок ниже.

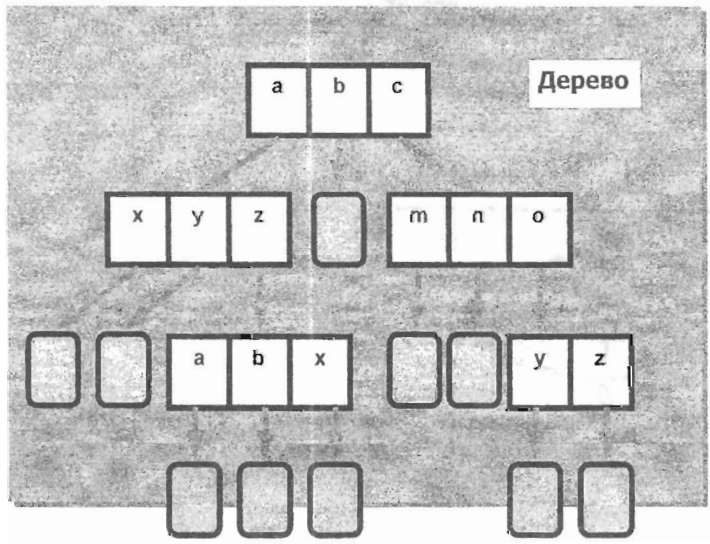


Рис.8.3 Иерархическая каталоговая система

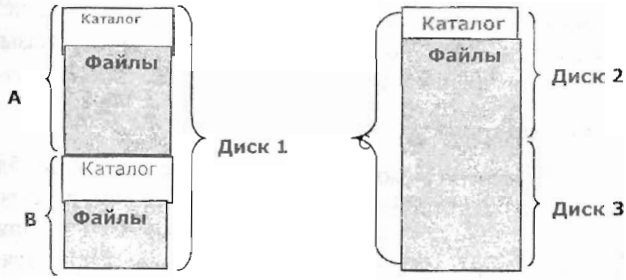


Рис. 8.4 Типичная структура организации файловой системы.

8.3.Структура и примеры файловых систем

Разработчикам файловых систем приходится заботиться о том, как файлам выделяется место на диске, как система контролирует размещение блоком в различных файлах. Различные варианты реализации файлов включают в себя непрерывные файлы, связные списки, таблицы размещения файлов. Различные файловые системы отличаются друг от друга, например, каталоговой структурой, управлением свободным дисковым пространством, способом учета принадлежности блоков файлам. Ниже приведены примеры наиболее распространенных файловых систем:

- Файловая система *CD-ROM.*
- Файловая система *CP/M.*
- Файловая система *MS DOS.*
- Файловая система *Windows.*
- Файловая система *UNIX.*

9. Ввод-вывод

Управление всеми устройствами ввода-вывода компьютера – одна из основных функций операционной системы. Операционная система должна предоставлять удобный и простой интерфейс между устройствами и остальными частями операционной системы.

9.1. Принципы аппаратуры ввода-вывода

Инженеры-электронщики рассматривают устройства ввода-вывода с точки зрения микросхем, проводов, источников питания и прочих физических компонент, из которых и состоит аппаратура. Программисты в первую очередь обращают внимание на интерфейс, предоставляемый программному обеспечению – команды, принимаемые аппаратурой, выполняемые ею функции и ошибки, о которых аппаратура может сообщить. Нас интересует именно программирование устройств ввода-

вывода, а не их проектирование, построение или поддержка. В то же время программирование многих устройств ввода-вывода часто оказывается тесно связанным с их внутренним функционированием.

9.1.1 Устройства ввода-вывода

Устройства ввода-вывода можно разделить на две категории: блочные устройства и символьные устройства. *Блочными* называются устройства, хранящие информацию в виде блоков фиксированного размера, причем у каждого блока имеется адрес. Обычно размеры блоков варьируются от 521 до 32 768 байт. Важное свойство блочного устройства состоит в том, что каждый его блок может быть прочитан независимо от остальных блоков. Наиболее распространенными блочными устройствами являются диски.

Другой тип устройств ввода-вывода – символьные устройства. Символьное устройство принимает или предоставляет поток символов без какой-либо блочной структуры. Оно не является адресуемым и не выполняет операцию поиска. Принтеры, сетевые интерфейсные карты, мыши (для указания точки на экране) и большинство других устройств, не похожих на диски, можно рассматривать как *символьные устройства*.

9.1.2. Контроллеры устройств

Устройства ввода-вывода обычно состоят из механической части и электронной части. Электронный компонент устройства называется контроллером устройства или *адаптером*. В персональных компьютерах он представляет собой печатную плату, вставляемой в слот расширения. Механический компонент находится в самом устройстве.

Работа контроллера заключается в конвертировании последовательного потока битов в блок байтов и выполнение коррекции ошибок, если это необходимо. Обычно байтовый блок собирается бит за битом в буфере контроллера. Затем проверяется контрольная сумма блока, и, если она совпадает с указанной в заголовке сектора, блок объявляется считанным без ошибок, после чего он копируется в оперативную память.

Контроллер монитора (*видеоадаптер*) также работает как бит-последовательное устройство. Он считывает в памяти байты, содержащие символы, которые следует отобразить и формирует сигналы, используемые для модуляции луча электронной трубки, заставляющие ее выводить изображение на экран. Видеоадаптер также формирует сигналы, управляющие горизонтальным и вертикальным возвратом электронного луча. Если бы ни контроллер, программисту пришлось бы управлять перемещениями аналогового электронного луча. В действительности же операционная система всего лишь инициализирует контроллер, задавая небольшое число параметров, таких, как количество символов или пикселов в строке и число строк на экране, а всю тяжелую работу по управлению передвижениями электронного луча по экрану выполняет контроллер.

9.1.3. Отображаемый на адресное пространство памяти ввод-вывод

У каждого контроллера есть несколько регистров, с помощью которых с ним может общаться центральный процессор. При помощи записи в эти регистры операционная система велит устройству предоставить данные, принять данные, включиться или выключиться и т. п. Читая из этих регистров, операционная система может узнать состояние устройства, например, готово ли оно к приему новой команды и т. д.

Помимо управляющих регистров, у многих устройств есть буфер данных, из которого операционная система может читать данные, а также писать данные в него. Например, для отображения пикселов на экране данные обычно помещаются в видеопамять, являющуюся, по сути, буфером данных, доступным операционной и другим программам для чтения и записи.

Достоинства данного метода:

- при этой схеме для обращения к устройствам ввода-вывода не требуются специальные команды процессора. В результате, программу, общающуюся с таким устройством, можно написать целиком на языке *C* или *C++*, без вставок на ассемблере или обращений к подпрограммам, написанным на ассемблере.
- при отображении регистров ввода-вывода на память не требуется специального механизма защиты от пользовательских процессов, пытающихся обращаться к внешним устройствам пространства пользователей.
- при отображении регистров ввода-вывода на память каждая команда процессора, обращающаяся к памяти, может с тем же успехом обращаться к управляющим регистрам устройства.

Отображение регистров ввода-вывода на память обладает недостатками. Во-первых, в большинстве современных компьютеров применяется *кэширование* памяти. Кэширование управляющих регистров привело бы просто к катастрофе. Чтобы не допустить такой ситуации, необходима специальная аппаратура, способная выборочно запрещать кэширование, например, в зависимости от номера страницы памяти, к которой обращается процессор. Таким образом, отображение регистров ввода-вывода на память увеличивает сложность аппаратуры и операционной системы, которой приходится управлять избирательным кэшированием.

Во-вторых, при едином адресном пространстве все модули памяти и все устройства ввода-вывода должны изучать все обращения процессора к памяти, чтобы определить, на которые им следует реагировать.

9.2. Принципы и задачи программного обеспечения ввода-вывода

Ключевая концепция разработки программного обеспечения ввода-вывода известна как независимость от устройств. Эта концепция означает возможность написания программ, способных получать доступ к любому устройству ввода-вывода без предварительного указания конкретного устройства. Все проблемы, связанные с отличиями этих устройств, должна решать операционная система.

Другим важным аспектом программного обеспечения ввода-вывода является обработка ошибок. Ошибки должны обрабатываться как можно ближе к аппаратуре. Если контроллер обнаружил ошибку чтения, он должен попытаться по возможности исправить эту ошибку сам. Если он не может это сделать, тогда эту ошибку должен обработать драйвер устройства, возможно, попытавшись прочитать этот блок еще раз.

Еще одним ключевым вопросом является способ переноса данных: синхронный (блокирующий) и асинхронный (управляемый прерываниями). Большинство операций ввода-вывода на физическом уровне являются асинхронными – центральный процессор запускает перенос данных и отправляется заниматься чем-либо другим, пока не придет прерывание. Тем, чтобы операции ввода-вывода, в действительности являющиеся асинхронными, выглядели как блокирующие в программах пользователя, занимается операционная система.

Еще одним аспектом программного обеспечения ввода-вывода является *буферизация*. Часто данные, поступающие с устройства, не могут быть сохранены сразу там, куда они в конечном итоге направляются. Например, когда пакет приходит по сети, операционная система не знает, куда его поместить, пока не будет изучено его содержимое, для чего этот пакет нужно где-то временно сохранить. Кроме того, для многих устройств реального времени крайне важными оказываются параметры сроков поступления данных (например, для устройств воспроизведения цифрового звука), поэтому полученные данные должны быть помещены в выходной буфер заранее, чтобы скорость, с которой эти данные получаются из буфера воспроизводящей программой, не зависела от скорости заполнения буфера. Таким образом, удается избежать неравномерности воспроизведения звука. Буферизация включает копирование данных в значительных количествах, что часто является основным фактором снижения производительности операций ввода-вывода.

Важным является понятие выделенных устройств и устройств коллективного использования. С некоторыми устройствами ввода-вывода, такими, как диски, может одновременно работать большое количество пользователей. При этом не должно возникать проблем, если несколько пользователей на одном и том же диске одновременно откроют файлы. Введение понятия выделенных (монопольно используемых) устройств также привносит целый спектр проблем, например такие, как взаимоблокировки. Тем не менее, операционная система должна уметь управлять как устройствами общего доступа, так и выделенными устройствами, позволяя избегать различных потенциальных проблем.

9.2.1 Программный ввод-вывод

Существует три фундаментально различных способа осуществления операций ввода-вывода: программный ввод-вывод, управляемый прерываниями ввод-вывод и ввод-вывод с использованием *DMA*.

Простейший вид ввода-вывода состоит в том, что всю работу выполняет центральный процессор. Этот метод называется программным вводом-выводом.

9.2.2. Управляемый прерываниями ввод-вывод

Предоставить центральному процессору возможность делать что-нибудь в то время, когда принтер, например, переходит в состояние готовности, можно при помощи прерываний. Когда выполняется системный вызов печати строки, буфер копируется в пространство ядра и первый символ строки копируется на принтер, как только принтер выставит бит готовности. После этого центральный процессор вызывает планировщик, который запускает какой-либо другой процесс. Процесс, попросивший распечатать строку, оказывается заблокирован на весь период печати строки.

9.2.3. Ввод-вывод с использованием DMA

Очевидный недостаток управляемого прерываниями ввода-вывода состоит в том, что прерывания происходят при печати каждого символа. Обработка прерываний занимает определенное время, поэтому такая схема не является эффективной. Решение этой проблемы заключается в использовании *DMA*. Идея состоит в том, чтобы позволить контроллеру *DMA* поставлять принтеру символы по одному, не беспокоя при этом центральный процессор. По существу, этот метод почти не отличается от программного ввода-вывода, с той лишь разницей, что всю работу вместо центрального процессора выполняет контроллер *DMA*.

Если контроллер *DMA* не может поддерживать полную скорость ввода или вывода с внешнего устройства, либо у центрального процессора нет других задач во время ожидания прерывания от *DMA*, тогда оба предыдущих метода ввода-вывода (программный и управляемый прерываниями) будут предпочтительнее.

9.3. Программные уровни ввода-вывода

Программное обеспечение ввода-вывода обычно организуется в виде четырех уровней, у каждого уровня есть четко очерченная функция, которую он должен выполнять, и строго определенный интерфейс с соседними уровнями. Функции и интерфейсы уровней меняются от одной операционной системы к другой, поэтому последующее рассмотрение всех уровней, начиная с нижнего, не является специфичным для какой-либо конкретной машины.

9.3.1. Обработчики прерываний

Прерывания должны быть скрыты в операционной системею. Лучший способ заключается в блокировке драйвера, начавшего операцию ввода-вывода, вплоть до окончания этой операции и получения прерывания.

Драйвер может заблокировать себя сам, выполнив на семафоре процедуру *down*, процедуру *wait* на переменной состояния, процедуру *receive* на сообщении.

Когда происходит прерывание, начинает работу обработчик прерываний. По окончании необходимой работы он может разблокировать драйвер, запустивший его. В некоторых случаях используется выполнение процедуры *up* на семафоре.

В других случаях обработчик прерываний вызывает процедуру монитора *signal* с переменной состояния. В третьем случае он посылает заблокированному драйверу сообщение. В любом случае драйвер разблокируется обработчиком прерываний. Эта схема лучше всего работает в драйверах, являющихся процессами ядра со своим собственным состоянием, стеком и счетчиком команд.

9.3.2. Драйверы устройств

У каждого контроллера есть набор регистров, используемых для того, чтобы давать управляемому им устройству команды и читать состояние устройства. Число таких регистров и команды, выдаваемые устройствам, зависят от конкретного устройства. Например, драйвер диска должен знать о секторах, дорожках, цилиндрах, головках, их перемещении и времени установки, двигателях и тому подобных вещах, необходимых для правильной работы диска. Очевидно, что драйверы диска и мыши будут сильно различаться.

Поэтому для управления каждым устройством ввода-вывода, подключенным к компьютеру, требуется специальная программа. Эта программа, называемая драйвером устройства, обычно пишется производителем устройства и распространяется вместе с устройством. Поскольку для каждой операционной системы требуются специальные драйверы, производители устройств обычно поставляют драйверы для нескольких наиболее популярных операционных систем.

Чтобы получить доступ к аппаратной части устройства, то есть к регистрам контроллера, драйвер устройства должен быть частью ядра операционной системы, по крайней мере, в существующих на сегодняшний день архитектурах. Современные операционные системы предполагают работу драйверов в ядре, мы рассмотрим здесь именно такую модель.

Так как в операционную систему будут устанавливаться куски программ (драйверы), написанные другими программистами, необходима определенная архитектура, позволяющая подобную установку. Это означает, что должна быть выработана строго определенная модель функций драйвера и его взаимодействия с остальной операционной системой. Драйверы устройств обычно располагаются под остальной операционной системой.

У драйвера устройства есть несколько функций. Наиболее очевидная функция драйвера состоит в обработке абстрактных запросов чтения и записи независимого от устройств программного обеспечения, расположенного над ними. Но кроме этого, они должны также выполнять еще несколько функций. Например, драйвер должен при необходимости инициализировать устройство. Ему также может понадобиться управлять энергопотреблением устройства и регистрацией событий.

Многие драйверы устройств обладают сходной общей структурой. Типичный драйвер начинает с проверки входных параметров. Если они не удовлетворяют определенным критериям, драйвер возвращает ошибку. В противном случае драйвер преобразует абстрактные термины в конкретные. Например, дисковый драйвер может преобразовывать линейный номер блока в номера головки, дорожки и секторы.

Затем драйвер может проверить, не используется ли это устройство в данный момент. Если устройство занято, запрос может быть поставлен в очередь. Если устройство свободно, проверяется аппаратный статус устройства, чтобы понять, может ли запрос быть обслужен прямо сейчас. Может оказаться необходимым включить устройство или запустить двигатель, прежде чем начнется перенос данных. Как только устройство включено и готово, может начинаться собственно управление устройством. Управление устройством подразумевает выдачу ему серии команд. Именно в драйвере определяется последовательность команд в зависимости от того, что должно быть сделано. Определившись с командами, драйвер начинает записывать их в регистры контроллера устройства.

После того как драйвер передал все команды контроллеру, ситуация может развиваться по двум сценариям. Во многих случаях драйвер устройства должен ждать, пока контроллер не выполнит для него определенную работу, поэтому он блокируется до тех пор, пока прерывание от устройства его не разблокирует. В других случаях операция завершается без задержек и драйверу не нужно блокироваться.

9.3.3. Независимое от устройств программное обеспечение ввода-вывода

Часть программного обеспечения ввода-вывода предназначена для работы с конкретными устройствами, другая часть является независимой от устройств. Расположение точной границы между драйверами и независимым от устройств программным обеспечением зависит от системы и устройств, так как некоторые функции могут быть выполнены независимым от устройств. Основная задача независимого от устройств программного обеспечения состоит в выполнении функций ввода-вывода общих для всех устройств и предоставлении единообразного интерфейса для программ уровня пользователя. Ниже рассмотрим некоторые из этих вопросов более подробно.

9.3.4. Единообразный интерфейс для драйверов устройств

Главный вопрос операционной системы – как сделать так, чтобы все устройства ввода-вывода и драйверы выглядели по возможности одинаково. Этот вопрос связан с интерфейсом между драйверами устройств и остальной операционной системой.

При этом значительно легче установить новый драйвер, при условии, что он соответствует стандартному интерфейсу. На практике не все устройства являются абсолютно идентичными, но обычно имеется небольшое число типов устройств, достаточно сходных друг с другом. Например, даже у блочных и символьных устройств есть много общих функций. Другой аспект единообразного интерфейса состоит в именовании устройств ввода-вывода. Независимое от устройств программное обеспечение занимается отображением символьных имен устройств на соответствующие драйверы.

9.3.5. Буферизация

Буферизация является важным вопросом как для блочных, так и для символьных устройств по самым разным причинам. Рассмотрим процесс, который хочет прочитать данные с модема. Одна из возможных стратегий обработки поступающих символов состоит в обращении процесса пользователя к системному вызову *read* и блокировке в ожидании отдельного символа. Каждый прибывающий символ вызывает прерывание. Процедура обработки прерываний передает символ пользовательскому процессу и разблокирует его. Поместив куда-либо полученный символ, процесс читает следующий символ, опять блокируясь.

Недостаток такого подхода состоит в том, что процесс пользователя должен быть активирован при прибытии каждого символа, что крайне неэффективно.

Улучшенный вариант: пользовательский процесс предоставляет буфер размером в *n* символов в пространстве пользователя, после чего выполняет чтение *n* символов. Процедура обработки прерываний помещает приходящие символы в буфер до тех пор, пока он не заполнится. Затем она активизирует процесс пользователя. Такая схема гораздо эффективнее предыдущей.

9.3.6. Захват и освобождение выделенных устройств

Некоторые устройства, например, устройство записи компакт-дисков, могут использоваться в каждый момент времени только одним пользователем. Операционная система должна рассмотреть запросы на использование этого устройства и либо принять их, либо отказать в выполнении запроса, в зависимости от доступности запрашиваемого устройства. Простой способ обработки этих запросов состоит в требовании к процессам обращаться напрямую к системному вызову open по отношению к специальным файлам для данных устройств. Если устройство недоступно, системный вызов *open* завершится неуспешно. Обращение к системному вызову *close* освобождает устройство.

Альтернативный подход состоит в предоставлении специального механизма для запроса и освобождения выделенных устройств. Попытка захватить недоступное устройство вызовет блокировку вызывающего процесса вместо возврата с ошибкой. Блокированные процессы устанавливаются в очередь. Раньше или позже запрашиваемое устройство освобождается и первому процессу в очереди разрешается захватить его и продолжить выполнение.

9.3.7. Независимый от устройств размер блока

У различных дисков могут быть разные размеры сектора. Независимое от устройств программное обеспечение должно скрывать этот факт от верхних уровней и предоставлять им единообразный размер блока, например, объединяя несколько физпческих сегментов в один логический блок. При этом более высокие уровни имеют дело только с абстрактными устройствами, с одним и тем же размером логического блока, не зависящим от размера физического сектора.

9.4. Программное обеспечение ввода-вывода пространства пользователя

Большая часть программного обеспечения ввода-вывода находится в операционной системе, а небольшие его порции состоят из библиотек, присоединенных к программам пользователя или даже целых программ, работающих вне ядра. Системные вызовы, включая системные вызовы ввода-вывода, обычно состоят из библиотечных процедур. Набор всех этих библиотечных процедур, несомненно, является частью системы ввода-вывода.

Не все программное обеспечение ввода-вывода пространства пользователя состоит из библиотечных процедур. Другую важную категорию составляет система спулинг (*spooling* – подкачка, предварительное накопление данных), которая представляет собой способ работы с выделенными устройствами в многозадачной системе. Типичное устройство, на котором используется спулинг: принтер. В принципе можно разрешить каждому пользователю открывать специальный символьный файл принтера, однако может быть так, что процесс открыл его, а затем не обращался к принтеру в течение нескольких часов. Ни один другой процесс в это время не сможет ничего напечатать.

Вместо этого создается специальный процесс, называемый *демоном*, и специальный каталог, называемый каталогом спулинга, чтобы распечатать файл, процесс сначала создает специальный файл, предназначенный для печати, который помещает в каталог спулинга. Этот файл печатает демон, единственный процесс, которому разрешается пользоваться специальным файлом принтера. Таким образом, потенциальная проблема, связанная с тем, что какой-либо процесс на слишком долгий срок захватит принтер, решается при помощи защиты специального файла принтера от прямого доступа пользователей.

9.5. Диски

Рассмотрим теперь некоторые реальные устройства ввода-вывода, к которым относятся диски. Затем рассмотрим таймер, клавиатуру и дисплей.

9.5.1. Аппаратная часть дисков

Существует множество типов дисков. Наиболее часто распространены магнитные диски (жесткие и гибкие), особенностью которых является одинаковая скорость чтения и записи, что делает их идеальными в качестве дополнительной памяти (страничная подкачка файлов, файловые системы и т.д.). Иногда, с целью создания высоконадежного устройства хранения, используются наборы жестких магнитных дисков. Для распространения программ, данных и фильмов используются различные виды оптических дисков *(CD-ROM, CD-R, CD-RW и DVD)*, запоминающее устройство *USB*.

В следующих разделах мы сначала познакомимся с аппаратной частью, а затем с программным обеспечением для этих устройств.

Магнитные диски организованы в цилиндры, каждый из которых содержит столько дорожек, сколько есть у устройства головок, установленных вертикально. Дорожки делятся на сектора, их количество обычно варьируется от 8 до 32 у гибких дисков и до нескольких сот. Многие контроллеры жестких дисков также могут совмещать операцию чтения или записи на одном диске с поиском на другом или даже нескольких дисках. Однако контроллеры гибких дисков не могут одновременно читать и писать на двух дисководах. (Чтение или запись требует от контроллера перемещения битов с максимальной скоростью, на которую он рассчитан, поэтому операция чтения или записи использует большую часть его вычислительных мощностей.) В жестких дисках с их встроенными микроконтроллерами несколько жестких дисков могут одновременно работать в одной системе, по крайней мере, переносить данные между диском и буфером контроллера. Однако между контроллером и оперативной памятью в каждый момент времени может происходить только одна операция по переносу данных. Способность одновременного выполнения двух или более дисковых операций может существенно снизить среднее время доступа.

Новые устройства RAID. Производительность центральных процессоров за последнее десятилетие увеличилась экспоненциально, удваиваясь примерно каждые 18 месяцев.

Для увеличения производительности центрального процессора все больше используются параллельные вычисления. Эти идеи были быстро приняты промышленностью, в результате чего был разработан новый класс устройств ввода-вывода, названный *RAID*. Паттерсон с соавторами определили *RAID* как *Redundant Array of Inexpensive Disks* – массив недорогих дисков с избыточностью, но промышленники переопределили букву *I* как *«Independent»* (независимые).

Идея, лежащая в основе системы *RAID*, состоит в том, что на компьютер (обычно большой сервер) устанавливается коробка, полная дисков. Вместо обычного дискового контролера устанавливается специальный *RAID*-контроллер, операционная система воспринимает их как один большой диск. Таким образом, для использования системы *RAID* не требуется никаких изменений программного обеспечения, что является большим плюсом с точки зрения системных администраторов.

Еще одно свойство всех *RAID*-систем состоит в том, что данные распределяются по дискам, а это позволяет распараллеливать операции.

Компакт-диски. В 70-е годы появились оптические диски (в противоположность магнитным). Их плотность записи сразу значительно превзошла плотность записи магнитных дисков. Изначально оптические диски были разработаны для записи телевизионных программ, но затем для них были придуманы другие сферы применения, в том числе хранение компьютерных данных. Благодаря их потенциально огромной емкости, оптические диски стали предметом большого числа исследований и совершили невероятно быструю эволюцию.

Для производства компакт-дисков используется мощный инфракрасный лазер, которым прожигаются отверстия диаметром 0,8 мкм в металлическом покрытии стеклянного диска-оригинала, называемого также мастер-диском. При воспроизведении полупроводниковый лазер низкой мощности освещает питы и участки между ними лучом

Последовательность нитов и промежутков между ними записывается в виде единой непрерывной спиральной дорожки, начинающейся недалеко от центра диска и заканчивающейся у края диска:

Компакт-диски с возможностью записи. Эти носители известны под названием *CD-R (CD-Recordable* – компакт-диски с возможностью записи).

Физически компакт-диски с возможностью записи, так же как и обычные *CD-ROM*, состоят из 120-мм поликарбонатных пластин, с той разницей, что на них нанесена спиральная дорожка глубиной 0,6 мм для направления луча лазера при записи *CD-R*.

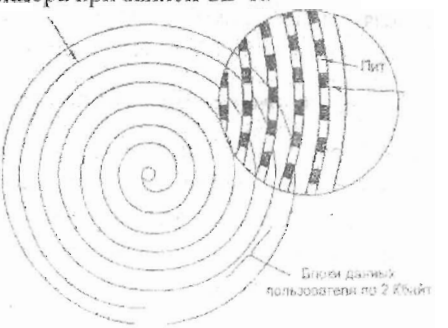


Рис. 9.1. Структура записи компакт-диска или *CD-ROM*

Многократно перезаписываемые компакт-диски. Технология *CD-ROM* с возможностью многократной перезаписи появилась под названием *CD-RW (CD-ReWritable* – многократно перезаписываемые компакт-диски). *CD-RW* по размерам ничем не отличаются от обычных компакт-дисков, *CD-ROM* или *CD-R*. Однако, вместо цианина или фталоцианина в *CD-RW* в качестве записывающего слоя используется сплав серебра, индия, сурьмы и теллура. У этого сплава два устойчивых состояния: кристаллический и аморфный, с различной отражающей способностью.

Экономически стало возможным создание оптических дисков большей емкости. В результате объединения технологических усилий и интересов трех богатых и мощнейших отраслей промышленности и была создана система *DVD (Digital Video Disk* цифровой видеодиск), теперь официально обозначает *Digital Versatile Disk* (универсальный цифровой диск). В *DVD* используется та же самая общая схема, что и для компакт-дисков, то есть те же 120-мм поликарбонатные диски, изготавливаемые печатным способом, содержащие углубления, освещаемые лазером и считываемые фотодетектором.

Эти усовершенствования позволили увеличить емкость в семь раз, то есть с 650 Мбайт до 4,7 Гбайт. Односкоростное устройство чтения *DVD* считывает данные со скоростью 1,4 Мбайт/с.

9.5.2 Форматирование дисков

Жесткий диск состоит из стопки сделанных из алюминия, стекла или других материалов пластин диаметра 5,25 дюйма или 3,5 дюйма.

Прежде чем диском можно будет пользоваться, каждой пластине нужно программно придать низкоуровневый формат. Этот формат состоит из серий концентрических дорожек, содержащих определенное число секторов, с короткими промежутками между секторами.

Заголовок начинается с определенной последовательности битов, обеспечивающей распознавание аппаратурой начала сектора. Он также содержит номера цилиндра и сектора и еще некоторую информацию. Размер порции данных определяется программой низкоуровневого форматирования. В большинстве дисков используются 512-байтовые секторы.

9.5.3. Алгоритмы планирования перемещения головок

В этом разделе мы рассмотрим некоторые вопросы, касающиеся в основном дисковых драйверов. Прежде всего, определим, сколько времени занимает считывание или запись одного блока диска.

Для большинства дисков время поиска существенно превосходит остальные функции, поэтому значительного увеличения производительности системы можно добиться, снижая время поиска.

9.5.4 Обработка ошибок

Производители дисков постоянно раздвигают технологические пределы, увеличивая линейную плотность битов. К сожалению, создать диск с высокими характеристиками без дефектов невозможно.

Дефектные блоки могут обрабатываться контроллером или операционной системой.

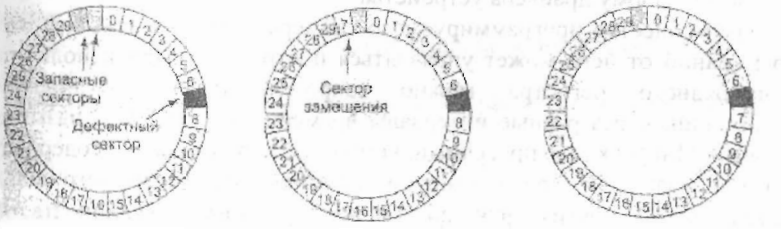


Рис. 9.2. Дорожка диска с дефектным сектором (а); замена дефектного сектора запасным (б); сдвиг всех секторов (в)

Ошибки могут возникать при выполнении обычных операций после установки диска. Если контроллер не может прозрачно преобразовывать секторы, тогда этим должна программно заниматься операционная система. Один из способов состоит в создании секретного файла, включающего в себя все дефектные секторы.

9.5.5 Стабильное запоминающее устройство

Можно создать дисковую подсистему, обладающую следующими свойствами: получив команду записи, такое устройство либо корректно записывает данные, либо не делает ничего, не изменяя хранящиеся на нем данные. Подобная система называется стабильным запоминающим устройством и реализуется программно.

9.6. Таймеры

Таймеры (также называемые часами) очень важны для работы любой многозадачной системы. Среди многих других задач, они следят за временем суток, не позволяют одному процессу надолго занять центральный процессор. Программное обеспечение таймера может принимать форму драйвера устройства.

Преимущество программируемого таймера состоит в том, что частота прерываний от него может управляться программно. При использовании 32-разрядного регистра можно запрограммировать возникновение прерываний через равные интервалы времени от 2 до 8,6 с, называемые тиками. Микросхемы программируемых таймеров обычно содержат два или три независимо программируемых счетчика и помимо этого обладают целым рядом других функций (например, могут увеличивать, а не уменьшать значение счетчика, не инициировать прерываний и т. д.).

Чтобы показания таймера не терялись, пока питание компьютера выключено, часы большинства компьютеров питаются от аккумулятора. Показания часов считываются при загрузке операционной системы.

9.6.1. Программное обеспечение таймеров

Аппаратно таймер инициирует прерывания через определенные интервалы времени. Обязанности драйвера часов варьируются в зависимости от операционной системы, но обычными являются следующие функции:

1. Слежение за временем суток.
2. Запрещение процессам работать дольше, чем им разрешено.
3 Учет использования центрального процессора.
4. Обработка системного вызова *alarm*, инициированного процессом пользователя.
5. Поддержка следящих таймеров для операционной системы.
6. Наблюдение, анализ и сбор статистики.

Операционной системе также требуются таймеры. Они называются *сторожевыми таймерами*.

Механизм обработки сторожевых таймеров, используемый драйвером часов, тот же, что применяется для сигналов пользователя. Единственное отличие состоит в том, что когда таймер срабатывает, вместо подачи сигнала драйвер часов вызывает процедуру, предоставляемую обратившимся к нему процессом. Эта процедура является частью процесса.

9.6.2. «Мягкие» таймеры

Идея «мягких» таймеров позволяет избежать лишних прерываний. «Мягкие» таймеры устанавливаются и срабатывают с той скоростью, с которой выполняются входы в ядро по следующим причинам:

1. Системные вызовы.
2. Ошибки преобразования адреса *TLB*.

3. Отсутствие страницы памяти.
4. Прерывания ввода-вывода.
5. Временное отсутствие работы для центрального процессора .

9.7. Алфавитно-цифровые терминалы

На практике сегодня наиболее часто встречаются следующие три типа терминалов:

1. Автономные терминалы с последовательным интерфейсом *RS-232* для связи с мэйнфреймами.
2. Дисплеи персональных компьютеров с графическим интерфейсом пользователя.
3. Сетевые терминалы.

9.7.1 Технические средства терминалов с интерфейсом *RS-232*

Терминалы с интерфейсом *RS-232* представляют собой технические устройства, состоящие из клавиатуры и дисплея, общающиеся по последовательному интерфейсу. Эти терминалы соединяются с интерфейсной платой при помощи 9-коптактиого или 25-контактного разьема. В системе *UNIX* линии последовательной передачи имеют имена вроде */dev/tty1 или dev/tty2.* В системе *Windows* они обычно называются *COM1* и *COM2.*

Терминалы с интерфейсом *RS-232* все еще применяются на мэйнфреймах, иногда соединенные по телефонной линии через модем, этот тип устройств поддерживается всеми системами *UNIX.*

9.7.2. Программное обеспечение ввода

Клавиатура и дисплей являются почти независимыми устройствами, поэтому мы будем рассматривать их здесь по отдельности.

Основная задача клавиатурного драйвера состоит в сборе символов. Если каждое нажатие на клавишу вызывает прерывание, драйвер может получать введенный символ во время обработки прерывания.

Хотя клавиатура и дисплей являются логически раздельными устройствами, многие пользователи привыкли видеть только что введенные с клавиатуры символы, отображаемыми на экране. Отображением символов на экране занимается исключительно программное обеспечение. Этот процесс называется *эхопечатью.*

9.7.3 Программное обеспечение вывода

Терминальный вывод несколько проще ввода. По большей части компьютер посылает символы терминалу, который их отображает. Для каждого терминала выделяется выходной буфер. Эти буферы могут входить в тот же пул буферов, что и входные буферы или представлять собой выделенные буферы.

9.8. Графические интерфейсы пользователя

На персональных компьютерах могут использоваться символьные интерфейсы. В течение нескольких лет доминировала система *MS-DOS* с символьным интерфейсом. Однако теперь на большинстве персональных компьютеров используется графический интерфейс пользователя *(GUI, Graphical User Interface).*

Графический интерфейс пользователя состоит из четырех основных элементов – *windows, icons, menus, pointing device* – окна, пиктограммы, меню, указывающее устройство

Программное обеспечение графического интерфейса пользователя может быть реализовано либо на уровне пользователя, как это делается в семействе систем *UNIX,* либо включено в саму операционную систему, как в случае *Windows.*

9.8.1 Аппаратное обеспечение клавиатуры, мыши и дисплея персонального компьютера

Все современные персональные компьютеры оснащены клавиатурами и растровым графическим дисплеем с изображением, отображаемым в памяти компьютера. Эти компоненты составляют части самого компьютера. Однако в современном персональном компьютере клавиатура и экран являются полностью раздельными устройствами, каждое со своим собственным драйвером.

Связь с клавиатурой может осуществляться через последовательный порт, параллельный порт или порт *USB*. При нажатии любой клавиши центральный процессор прерывается и драйвер клавиатуры извлекает символ, читая порт ввода-вывода. Все остальное осуществляется программно, в основном в драйвере клавиатуры.

У большинства персональных компьютеров имеется мышь или, в некоторых случаях, шаровой манипулятор, представляющий собой обычную мышь, лежащую на спине. Наиболее распространенный тип компьютерных мышей содержит в себе резиновый шар, выглядывающий из отверстия в днище мыши и вращающийся, когда мышь двигается по столу или коврику, При вращении шар поворачивает прижатые к нему резиновые ролики, закрепленные на перпендикулярных осях. При движении мыши с запада на восток вращается ось *x*, а движение мыши с севера на юг заставляет вращаться ось *y*. Когда мышь преодолевает определенное минимальное расстояние, нажимается или отпускается одна из кнопок мыши, мышь посылает компьютеру сообщение. Обычно это минимальное расстояние составляет около 0,1 мм. У мыши может быть одна, две или три кнопки. Сообщение, посылаемое компьютеру, содержит три параметра; изменения позиции по координатам *x* и *y*, *dx* и *dy* и состояние кнопок. Формат сообщения зависит от системы и числа кнопок мыши. Обычно оно занимает 3 байта, большинство мышей способно передавать до 40 сообщений в секунду

Рассмотрим аппаратную часть дисплея. Дисплеи могут быть разделены на две основные категории: устройства с *векторной* графикой и устройства с *растровой* графикой. Векторные графические устройства могут выполнять такие команды, как вывод точек, рисование линий, геометрических фигур и текста. У растровых графических устройств, напротив, область вывода представляет собой прямоугольную сетку точек, называемых пикселами, каждая из которых может принимать различные значения яркости или цвета. Сейчас единственными векторными графическими устройствами остались *плоттеры*. Все остальные графические устройства используют растровую графику.

Растровые графические дисплеи реализуются при помощи специального устройства, называемого *графическим адаптером*. Графический адаптер содержит специальную память, называемую *видеопамятью* и образующую часть адресного пространства компьютера. Это означает, что нейтральный процессор обращается к ней так же, как и к остальной оперативной памяти. Здесь хранится образ экрана либо в символьном, либо в растровом виде. В символьном виде каждый байт (или два байта) видеопамяти содержат один отображаемый символ. В растровом виде каждый пиксел экрана представляется в видеопамяти отдельно. На каждый пиксел отводится от одного бита для простейшего черно-белого изображения, до 24 бит – для цветного дисплея высокого качества.

Кроме того, в состав графического адаптера входит микросхема, называющаяся *видеоконтроллером*. Эта микросхема получает символы из видеопамяти и формирует соответствующий им видеосигнал, посылаемый на монитор.

9.8.2. Программное обеспечение ввода

Получив символ, клавиатурный драйвер должен начать его обработку. Для достижения большей гибкости в настройке раскладок клавиатуры многие операционные системы предоставляют загружаемые кодовые страницы или, как их еще называют, карты клавиш. Они позволяют выбирать способ преобразования скан-кодов клавиш в символы, предоставляемые приложению, либо во время загрузке системы, либо позднее.

9.8.3. Программное обеспечение вывода для *Windows*

Программное обеспечение вывода для графического интерфейса пользователя представляет собой весьма обширную тему. Рассмотрим программный интерфейс приложения *Windows API*, поддерживаемый всеми 32-разрядными версиями системы *Windows*.

Поскольку после полной обработки сообщения система *Windows* уведомляется об этом, она может воздержаться от отправки следующего сообщения до того, как будет обработано предыдущее. Таким образом, удается избежать возникновения ситуации состязаний.

Программа, работающая в системе *Windows* обычно создает одно или несколько окон, для каждого из которых создается объект класса. В конечном итоге поведением программ управляют поступающие события, обрабатывающиеся специальными процедурами. Эта модель принципиально отличается от подхода, принятого в системе *UNIX*.

9.9. Сетевые терминалы

Сетевые терминалы используются для того, чтобы соединить удаленного пользователя с компьютером по локальной или глобальной сети.

9.9.1. Система *X Window*

Терминал, содержащий центральный процессор, такой же мощный, как и у основного компьютера, с мегабайтами памяти, клавиатурой и мышью представляет собой *X*-терминал, на котором работает система *X Window System*. *X*-терминал представляет собой компьютер, на котором работают *X*-программы и который взаимодействует с программами, работающими на удаленном компьютере).

Система *X Windows* представляет собой всего лишь оконную систему. Она не является полным графическим интерфейсом пользователя. Для предоставления пользователю полного графического интерфейса поверх системы *X Windows* запускаются другие уровни программного обеспечения.

Модульный дизайн, состоящий из нескольких уровней и большого количества программ, делает систему *X Windows* в высочайшей степени переносимой и гибкой. Она была установлена на большинство версий системы *UNIX*, включая *Sun Solaris*, *BSD*, *AIX*, *Linux* и т. п., что предоставило разработчикам стандартный интерфейс пользователя на различных платформах. Система *X Windows* была также установлена на другие операционные системы.

Как и *Windows*, система *X Windows* в значительной степени управляется событиями. Поток событий направляется от рабочей станции к программе, обычно в ответ на некое действие пользователя, например ввод с клавиатуры, перемещение мыши или открытие окна. Каждое сообщение имеет 32 байт в длину, из которых первый байт содержит тип сообщения, а остальные 31 байт содержат дополнительную информацию.

9.10 Управление режимом энергопотребления

Первый универсальный электронный компьютер *ENIAC (Electronic Numerical Integrator and Calculator* – электронный цифровой интегратор и калькулятор) состоял из 18 000 электронных ламп и потреблял 140 кВт. В результате счета за электричество были довольно высоки. После изобретения транзистора потребление электроэнергии снизилось на несколько порядков, в результате чего компьютерная промышленность потеряла интерес к экономии электроэнергии. Однако сегодня по

различным причинам управление режимом электропитания снова оказалось в центре внимания, и занимается этим в большой степени операционная система.

Обычный настольный персональный компьютер содержит 200-ваттный блок питания (КПД такого блока питания составляет около 85%. Это означает, что около 15% энергии преобразуется в тепло уже в блоке питания). Если одновременно включить питание 100 млн. таких машин по всему миру, то вместе они будут использовать около 20 ГВТ. Эта мощность примерно соответствует мощности 20 атомных электростанций среднего размера. Если бы потребление электроэнергии компьютерами можно было уменьшить вдвое, то мы могли бы закрыть 10 атомных электростанций. С точки зрения охраны окружающей среды закрытие 10 атомных электростанций (или эквивалентного количества электростанций, работающих на сжигаемом топливе) является большим достижением и за это стоит побороться.

Другая область, где потребление электроэнергии также является важным вопросом, – это переносные компьютеры, питаемые от батарей, включая ноутбуки, лэптопы, палмтопы и *web*-блокноты. Основой проблемы является то, что батареи не могут удерживать количество энергии, достаточное для долгой работы компьютера. Обычное время, на которое хватает заряда аккумуляторов, не превышает нескольких часов.

К снижению потребления электроэнергии существует два основных подхода. Первый из них заключается в выключении операционной системой тех частей компьютера (главным образом, устройств ввода-вывода), которые не используются в данный момент. Второй подход состоит в снижении потребления энергии прикладными программами, возможно, за счет снижения качества восприятия пользователем.

9.10.1. Аппаратный аспект

Электрические элементы подразделяются на два основных типа: одноразовые и перезаряжаемые (аккумуляторы).

Основной метод, предпринимаемый многими производителями компьютеров для продления жизни батарей, состоит в проектировании центрального процессора, памяти и устройств ввода-вывода, способных находиться в нескольких режимах энергопотребления: включенном, режиме ожидания, режиме глубокой «зимней спячки» и в выключенном состоянии.

9.10.2. Аспект операционной системы

Операционная система играет ключевую роль в управлении режимом энергопотребления. Она управляет всеми устройствами, поэтому ей нужно решать, какое устройство и когда выключать. Если выключаемое устройство потребуется снова в течение короткого интервала времени, выключения

устройства будут лишь приводить к раздражающим задержкам при его включении. С другой стороны, если операционная система будет слишком долго ждать, прежде чем выключить устройство, это приведет к напрасной растрате электроэнергии.

Суть состоит в том, чтобы найти алгоритмы, позволяющие операционной системе принимать правильные решения, касающиеся энергопотребления. Сложность в том, что понятие «правильного» решения в большой степени субъективно для разных пользователей.

Дисплей. Рассмотрим теперь устройства, являющиеся самыми крупными потребителями электроэнергии. Больше всех устройств электроэнергии потребляет дисплей. Можно задать режим пониженного энергопотребления, обратно дисплей может быть выведен в активное состояние практически мгновенно нажатием одной клавиши, при этом изображение регенерируется из видеопамяти.

Жесткий диск. Следующим главным потребителем является жесткий диск. Он потребляет значительное количество энергии для поддержания высокой скорости вращения даже при отсутствии обращений к нему. Многие компьютеры, особенно лэптопы, останавливают жесткий диск, если в течение определенного времени к нему нет обращений, когда он оказывается нужен, диск запускается снова. К сожалению, время раскрутки диска занимает несколько секунд, что составляет весьма ощутимую для пользователя задержку.

Кроме того, на раскрутку диска требуется дополнительная энергия. На практике, однако, в большинстве систем применяется консервативная политика выключения устройств после истечения его интервала времени, в течение которого к ним не было обращений.

Другой способ сохранения энергии диска заключается в поддержании большого дискового кэша в оперативной памяти. Аналогично, запись на диск также может буферизироваться в кэше. При этом диск может оставаться выключенным, пока не переполнится кэш или не понадобится блок, отсутствующий в кэше.

Оперативная память. Для энергосбережения при работе с памятью возможны два подхода. Во-первых, можно периодически сохранять и выключать кэш. Его всегда можно перезагрузить из оперативной памяти без потерь информации. Перезагрузка может выполняться автоматически и достаточно быстро, таким образом, данный вариант не является глубоким.

Более радикальный метод заключается в сохранении содержимого оперативной памяти на диске, после чего может быть выключена сама оперативная память.

Беспроводная связь. Последнее время появляется все большее число переносных компьютеров, соединенных с внешним миром (например, с Интернетом) при помощи беспроводной связи. Требующиеся для этого

радиопередатчики и радиоприемники часто оказываются энергоемкими.

Эффективное решение состоит в том, что мобильные компьютеры общаются с неподвижными базовыми станциями, обладающими большими объемами оперативной памяти и дискового пространства, а также не связанными ограничениями в потреблении энергии.

Исходящие сообщения, сформированные в период, когда радио отключено, буферизируются на мобильном компьютере. Если буферу угрожает переполнение, радиопередатчик включается и сообщения, стоящие в очереди, передаются на базовую станцию.

Интерфейс драйвера. В системе *Windows* имеется тщательно продуманный механизм управления энергопотреблением, называемый *ACPI (Advanced System Configuration and Power Interface* – усовершенствованный интерфейс конфигурирования системы и управления энергопотреблением). Операционная система может посылать любому драйверу, поддерживающему данный интерфейс, команды с требованием сообщить возможности его устройств и их текущие состояния. Эта способность особенно важна в комбинации с использованием стандарта *Plug and Play*, так как сразу после загрузки операционная система даже не знает, какие устройства присутствуют в компьютере, не говоря уже об их свойствах, касающихся энергопотребления или возможности управлять им.

Операционная система также может посылать драйверам команды, приказывая им снизить уровни энергопотребления.

Заключение

В представленном курсе «Операционные системы» рассмотрены многие аспекты функционирования операционных систем, но не все. Остальные разделы, например, такие, как мультимедийные операционные системы, многопроцессорные операционные системы, проблемы безопасности в операционных системах излагаются в расширенном курсе «Операционные системы».

ЛИТЕРАТУРА

1. Abraham Silberschatz, Peter Baer Galvin and Greg Gadne.
Operating system concepts.
(sixth edition)
Published by John Wiley & Sons, 2003

2. Andrew S. Tanenbaum.
Modern Operating Systems
(second edition)
Published by Prentice-Hall, 2001

3. W. Stallings. Operating Systems: Internals and Design Principles.
(fourth edition)
Published by Prentice-Hall, 2001.