Основные принципы управления ресурсами и организации доступа к этим ресурсам.

Управление ресурсами в операционной системе.

Важнейшей функцией ОС является организация рационального использования всех аппаратных и программных ресурсов ЭВМ.

К основным ресурсам могут быть отнесены:

- память
- внешние устройства
- данные и программы.
- Управление таким ресурсом, как процессор сводится к управлению процессами, которые в нем протекают.

Управление процессами.

- Важнейшей частью ОС, непосредственно влияющей на функционирование вычислительной машины, является подсистема управления процессами.
- Процесс (или по-другому, задача) - абстракция, описывающая выполняющуюся программу.

- Для операционной системы процесс представляет собой единицу работы, заявку на потребление системных ресурсов.
- Подсистема управления процессами планирует выполнение процессов, то
  есть распределяет процессорное время между несколькими одновременно
  существующими в системе процессами, а также занимается созданием и
  уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами, поддерживает взаимодействие между процессами.

Состояние процессов.

- В многозадачной системе процесс может находиться в одном из трех основных состояний:

Состояние	Характеристика состояния
Выполнение	Активное состояние процесса, во время которого процесс обладает всеми необходимыми ресурсами и непосредственно выполняется процессором
Ожидание	Пассивное состояние процесса, процесс заблокирован, он не может выполняться по своим внутренним причинам, он ждет осуществления некоторого события, например, завершения операции ввода-вывода, получения сообщения от другого процесса, освобождения какого-либо необходимого ему ресурса.
Готовность	Пассивное состояние процесса, процесс заблокирован в связи с внешними по отношению к нему обстоятельствами: процесс имеет все требуемые для него ресурсы, он готов выполняться, однако процессор занят выполнением другого процесса.

- В ходе жизненного цикла каждый процесс переходит из одного состояния в другое в соответствии с алгоритмом планирования процессов, реализуемым в данной ОС. Типичный граф состояний процесса показан на рисунке .

Рис. 17. Граф состояний процесса в многозадачной среде

В состоянии ВЫПОЛНЕНИЕ в однопроцессорной системе может находиться только один процесс, а в каждом из состояний ОЖИДАНИЕ и ГОТОВНОСТЬ - несколько процессов, эти процессы образуют очереди со - ответственно ожидающих и готовых процессов.

- Жизненный цикл процесса начинается с состояния ГОТОВНОСТЬ, когда процесс готов к выполнению и ждет своей очереди.
- При активизации процесс переходит в состояние ВЫПОЛНЕНИЕ и находится в нем до тех пор, пока либо он сам освободит процессор, перейдя в состояние ОЖИДАНИЯ какого-нибудь события, либо будет насильно "вытеснен" из процессора, например, вследствие исчерпания отведенного данному процессу кванта процессорного времени.
- В последнем случае процесс возвращается в состояние ГОТОВНОСТЬ. В это же состояние процесс переходит из состояния ОЖИДАНИЕ, после того, как ожидаемое событие произойдет.

Таким образом, на протяжении существования процесса его выполнение может быть многократно прервано и продолжено.
Для того, чтобы возобновить выполнение процесса, необходимо восстановить состояние его операционной среды (состояние операционной
среды отображается состоянием регистров, режимом работы процессора,
указателями на открытые файлы и т.д.). Эта информация называется контекстом процесса.

- Кроме этого, ОС для реализации планирования процессов требуется дополнительная информация: идентификатор процесса, состояние процесса, данные о степени привилегированности процесса и другая информация.
В некоторых ОС (например, в ОС UNIX) информацию такого рода, используемую ОС для планирования процессов, называют дескриптором
процесса.
Дескриптор процесса по сравнению с контекстом содержит более оперативную информацию, которая должна быть легко доступна подсистеме планирования процессов. Контекст процесса содержит менее акту Выполнение
Ожидание
Завершение

Рис. 17. Граф состояний процесса в многозадачной среде
актуальную информацию и используется ОС только после того, как принято
решение о возобновлении прерванного процесса.

- Алгоритмы планирования процессов

Планирование процессов включает в себя решение следующих задач:

- определение момента времени для смены выполняемого процесса;
- выбор процесса на выполнение из очереди готовых процессов;
- переключение контекстов "старого" и "нового" процессов.

Наиболее часто встречаются две группы алгоритмов планирования процессов:

- 1) алгоритмы, основанные на квантовании,
- 2) алгоритмы, основанные на приоритетах.

В соответствии с алгоритмами, основанными на квантовании, смена активного процесса происходит, если происходит одно из следующих событий: процесс завершился и покинул систему; произошла ошибка; процесс перешел в состояние ОЖИДАНИЕ; исчерпан квант процессорного
времени, отведенный данному процессу.

- Процесс, который исчерпал свой квант, переводится в состояние ГОТОВНОСТЬ и ожидает, когда ему будет предоставлен новый квант процессорного времени, а на выполнение в соответствии с определенным правилом выбирается новый процесс из очереди готовых.
- Таким образом, ни один процесс не занимает процессор надолго, поэтому квантование широко используется в системах разделения времени. Очередь готовых процессов может быть организована циклически, по правилу "первый пришел - первый обслужился" (FIFO) или по правилу "последний пришел - первый обслужился" (LIFO)

- Другая группа алгоритмов использует понятие "приоритет" процесса. Приоритет - это число, характеризующее степень привилегированности процесса при использовании ресурсов ЭВМ. Чем выше приоритет процесса, тем меньше времени он будет проводить в очередях. Приоритет может назначаться директивно администратором системы в зависимости от важности работы
- (рис. 18) либо вычисляться самой ОС по определенным правилам, он может оставаться фиксированным на протяжении всей жизни процесса либо изменяться во времени в соответствии с некоторым законом.

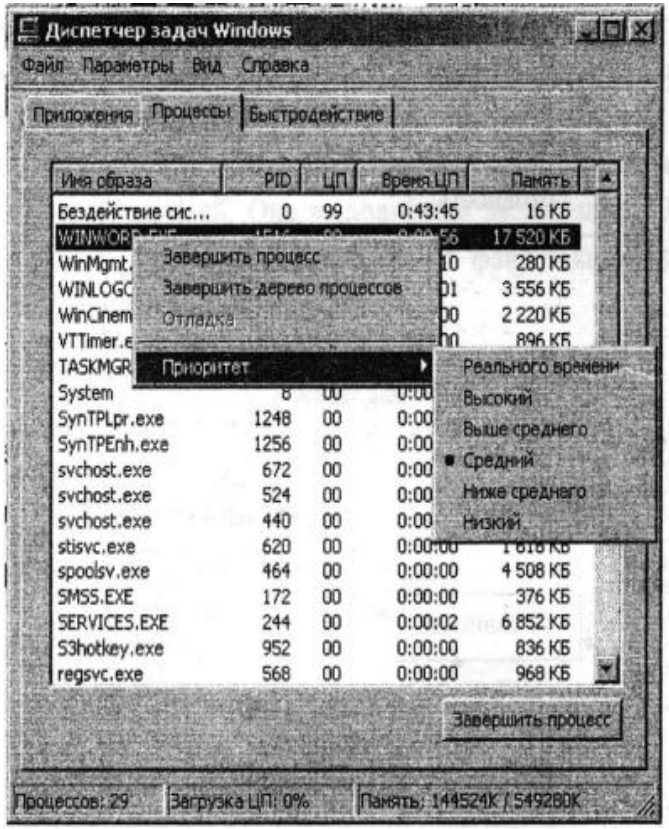


Рис. 18. Изменение приоритета процесса в ОС Windows 20

- Существует две разновидности приоритетных алгоритмов: алгоритмы, использующие относительные приоритеты; алгоритмы, использующие абсолютные приоритеты.
- В обоих случаях выбор процесса на выполнение из очереди готовых процессов осуществляется одинаково: выбирается процесс, имеющий наивысший приоритет. По разному решается проблема определения момента смены активного процесса. В системах с относительными приоритетами активный процесс выполняется до тех пор, пока он сам не покинет процессор, перейдя в состояние ОЖИДАНИЕ или завершив свое выполнение (рис. 19, а)

- В системах с абсолютными приоритетами выполнение активного процесса может быть прервано если в очереди готовых процессов появился процесс, приоритет которого выше приоритета активного процесса. В этом случае прерванный процесс переходит в состояние готовности (рис. 19, б)

Во многих операционных системах алгоритмы планирования построены с использованием как квантования, так и приоритетов. Например , в основе планирования лежит квантование, но величина кванта и/или порядок выбора процесса из очереди готовых определяется приоритетами процессов.

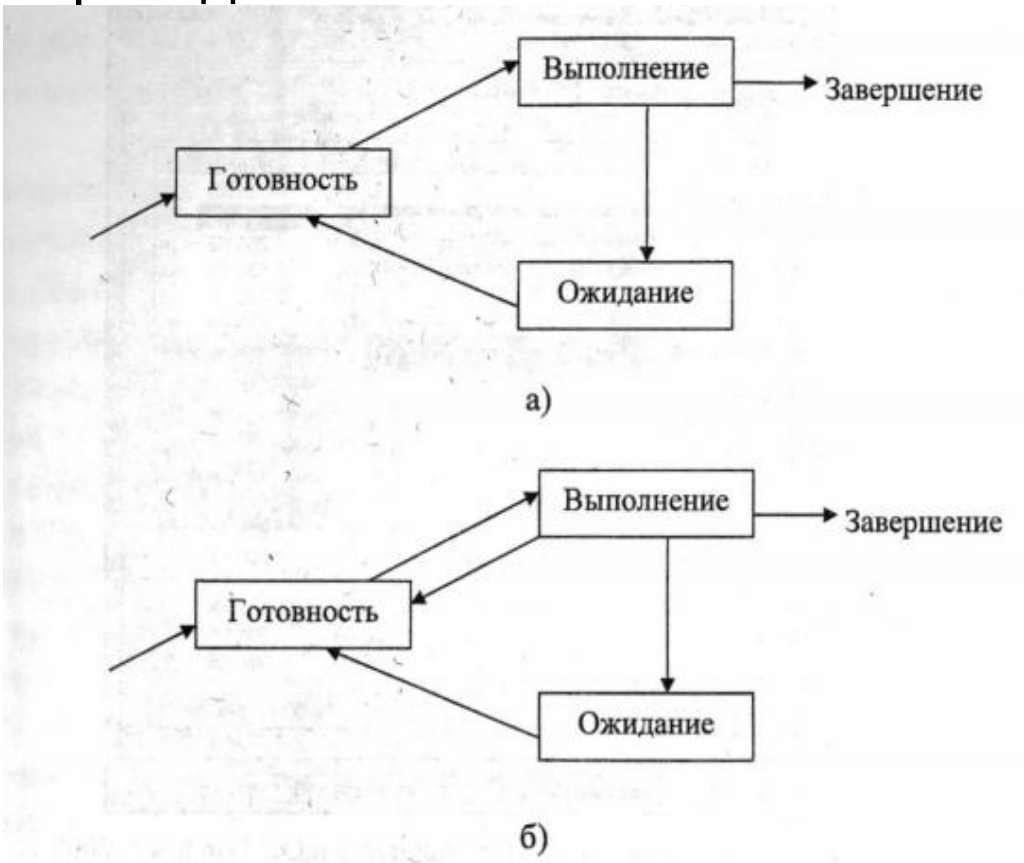


Рис. 19. Граф состояний процесса в многозадачной среде (а) – с относительными приоритетами; (б) – с абсолютными приоритетами

Взаимодействие процессов

- Процессам часто нужно взаимодействовать друг с другом, например,
один процесс может передавать данные другому процессу, или несколько
процессов могут обрабатывать данные из общего файла. Во всех этих случаях возникает необходимость синхронизации процессов.
Ситуации, когда два или более процессов обрабатывают разделяемые
данные, и конечный результат зависит от соотношения скоростей процессов, называются гонками.

- Важным понятием синхронизации процессов является понятие "критическая секция" программы. Критическая секция - это часть программы,
в которой осуществляется доступ к разделённому ресурсу. Например,
пусть выполняется два процесса (рис. 20). Они содержат одинаковую
часть, отвечающую за работу с одним и тем же разделённым ресурсом
(разделённый ресурс - файл заданий и переменная NEXT). Эта часть и называется критической секцией. Она выполняется за три шага (3 «команды»): прочитать NEXT, поместить имя файла в файл заказов, увеличить
значение NEXT на единицу.

Чтобы исключить эффект гонок, необходимо обеспечить, чтобы в каждый момент в критической секции, связанной с определенным ресурсом,
находился максимум один процесс. Этот прием называют взаимным исключением.

Существуют 3 подхода к решению проблемы синхронизации:

1. Реализация критических секций с использованием блокирующих переменных.

С каждым разделяемым ресурсом связывается переменная, которая принимает значение 1, если ресурс свободен, и значение 0, если ресурс занят.

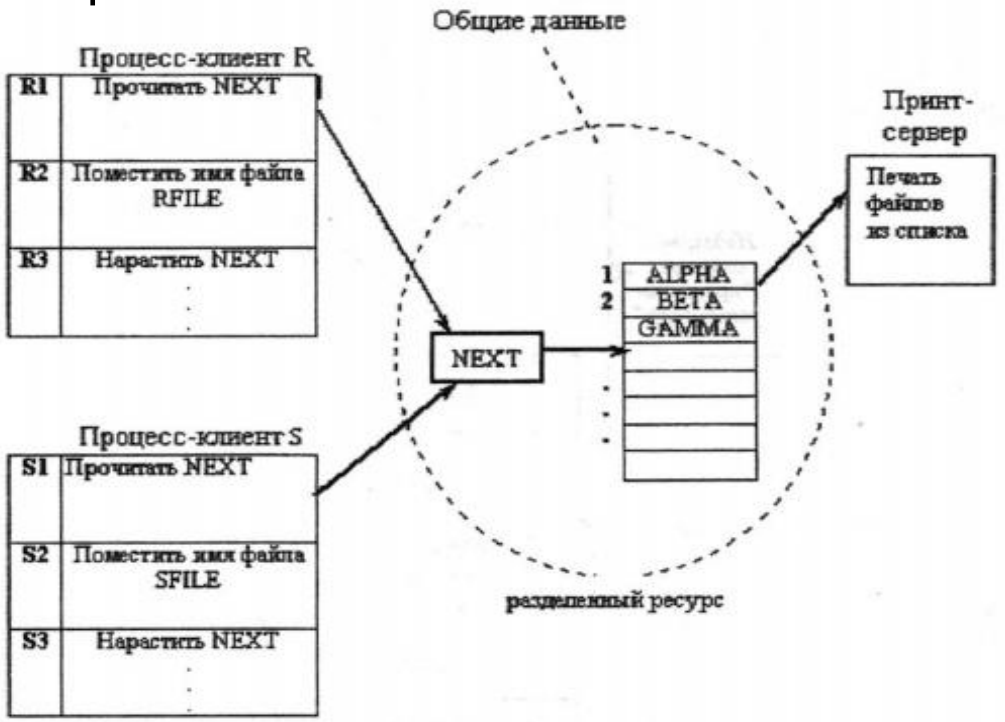
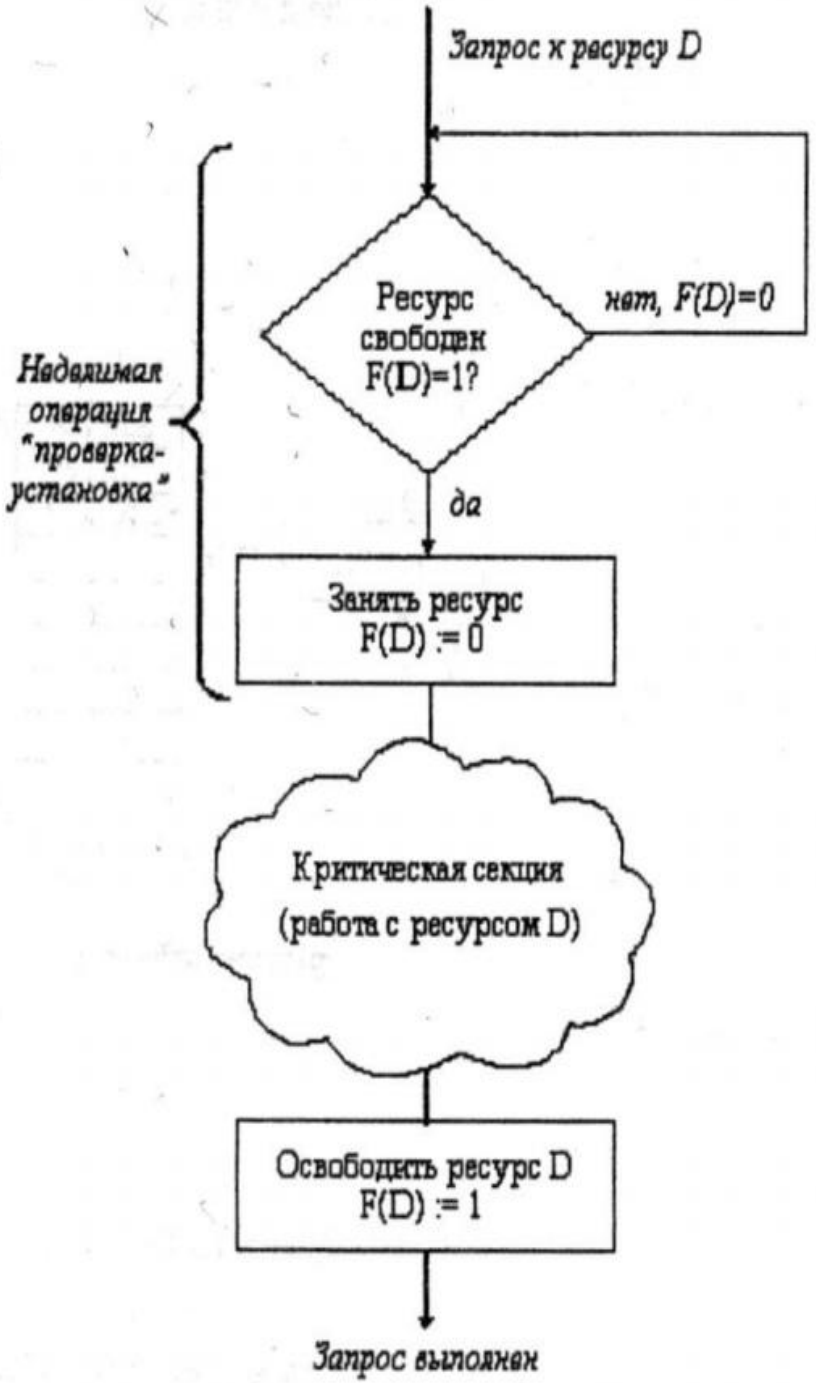


Рис. 20. Пример необходимости синхронизации

- На рисунке 21 показан фрагмент алгоритма процесса, использующего для реализации взаимного исключения доступа к разделяемому ресурсу D блокирующую переменную F(D). Реализация критических секций с использованием блокирующих переменных имеет существенный недостаток: в течение времени, когда один процесс находится в критической секции, другой процесс, которому требуется тот же ресурс, будет выполнять рутинные действия по опросу блокирующей переменной, бесполезно тратя процессорное время.



- 2. Реализация критических секций с использованием аппарата событий.
  С помощью аппарата событий решаются не только проблемы взаимного исключения, но и более общие задачи синхронизации процессов.
- В операционных системах для реализации аппарата событий используются две системные функции, которые условно назовем WAIT(x) и POST(x), где
  x - идентификатор ресурса.
- На рисунке 22 показан фрагмент алгоритма процесса, использующего эти функции. Если ресурс занят, то процесс не выполняет циклический опрос, а вызывает системную функцию WAIT(D).
- Функция WAIT(D) переводит этот активный процесс в состояние ОЖИДАНИЕ ресурса D и делает отметку в дескрипторе процесса о том, что процесс ожидает освобождения ресурса D.
- Процесс, который в это время использует ресурс D, после выхода из критической секции выполняет системную функцию POST(D), в результате чего операционная система просматривает очередь ожидающих процессов и переводит процесс, ожидающий освобождения ресурса D, в состояние ГОТОВНОСТЬ. В этом случае блокирующая переменная F(D) называется семафором.

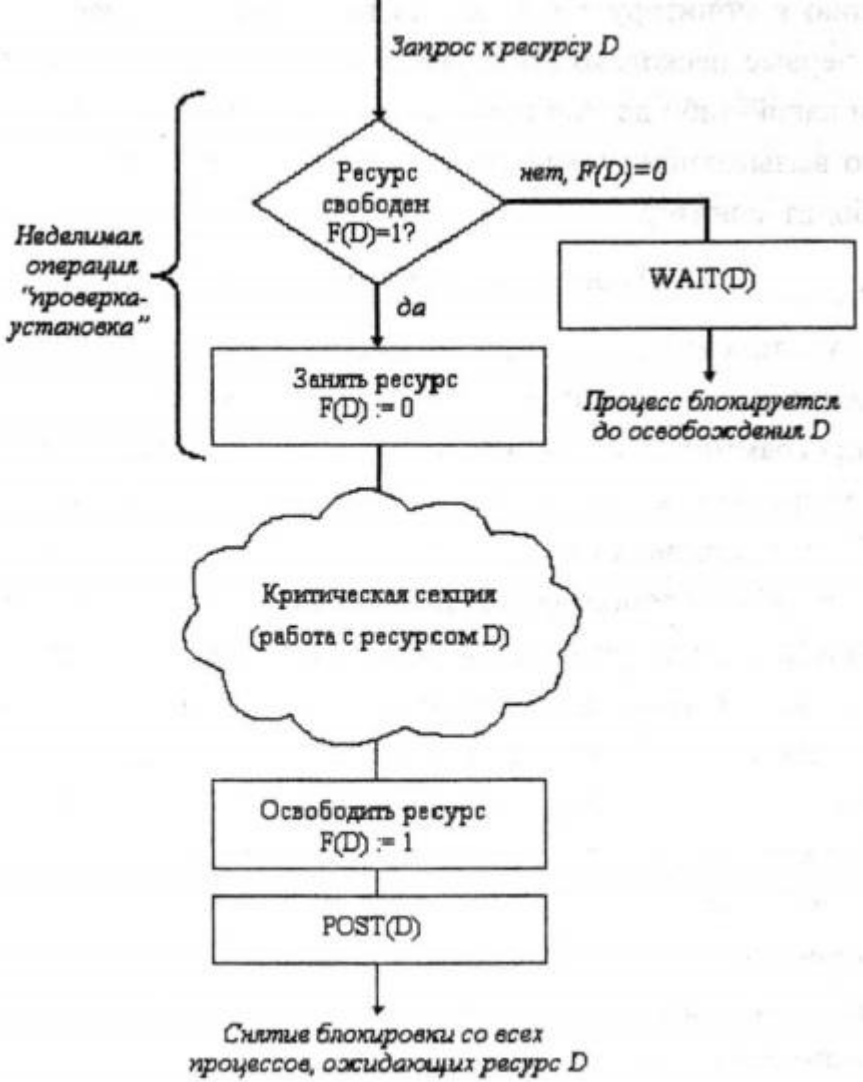


Рис. 22. Реализация критической секции с использованием аппарата событий

Достоинство синхронизации с использованием аппарата событий состоит в следующем: в течение времени, когда один процесс находится в критической секции, другой процесс, которому требуется тот же ресурс, не будет выполнять рутинные действия по опросу блокирующей переменной, бесполезно тратя процессорное время, то есть отсутствие активного ожидания представления ресурса.

Недостаток состоит в том, что при использовании семафоров может возникнуть ситуация взаимной блокировки, или тупика.

- 3. Реализация критических секций с использованием монитора.

- Монитор - это набор процедур, переменных и структур данных. Процессы могут вызывать процедуры монитора, но не имеют доступа к внутренним данным монитора и только один процесс может быть активным по отношению к монитору. Обычно, когда процесс вызывает процедуру монитора, первые несколько инструкций этой процедуры проверяют, не активен ли какой-либо другой процесс по отношению к этому монитору. Если да, то вызывающий процесс приостанавливается, пока другой процесс не освободит монитор.

Многонитевая организация процесса

- Для увеличения быстродействия процесса современные ОС предлагают использовать механизм многонитевой обработки (multithreading).
- Мультипрограммирование в этом случае реализуется на уровне нитей, и задача, оформленная в виде нескольких нитей в рамках одного процесса, может быть выполнена быстрее за счет псевдопараллельного (или параллельного в мультипроцессорной системе) выполнения ее отдельных частей.
- Такой подход к организации процесса также позволяет сделать более удобным пользовательский интерфейс многонитиевых программ.

Многонитевая организация процесса

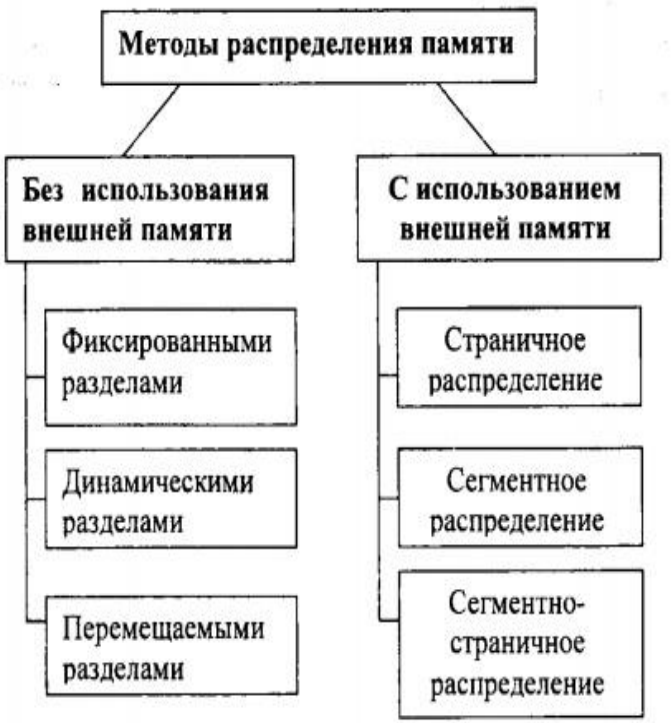
- Для увеличения быстродействия процесса современные ОС предлагают использовать механизм многонитевой обработки (multithreading).
- Мультипрограммирование в этом случае реализуется на уровне нитей, и задача, оформленная в виде нескольких нитей в рамках одного процесса, может быть выполнена быстрее за счет псевдопараллельного (или параллельного в мультипроцессорной системе) выполнения ее отдельных частей.
- Такой подход к организации процесса также позволяет сделать более удобным пользовательский интерфейс многонитиевых программ.

- Примером применения многонитиевой технологии при программировании может служить, например, программный продукт TOAD компании Quest Software. Данная программа предназначена для работы с системой управления базами данных (СУБД) ORACLE. Одной из основных функций данного продукта является выполнение запросов данных из базы данных. В общем случае выполнение запроса данных из СУБД представляет собой последовательность следующих действий: установление связи с СУБД, отправка в СУБД запроса, получение от СУБД результирующего набора данных (удовлетворяющих условиям запроса) и отображение этих данных в пользовательском интерфейсе программы.

- В зависимости от многих факторов (загрузки СУБД, сложности и корректности запроса, объема результирующих данных и т.д.) эти действия могут занимать достаточно большое время, в течение которого работа пользователя с программой будет заблокирована и процесс будет проверять, выполнились или нет определенные этапы получения от СУБД запрошенных данных. В связи с этим в программе TOAD предусмотрено использование многонитиевого режима, при котором действия по получению из СУБД данных для каждого запроса пользователя выполняются в пределах отдельных нитей общего процесса TOAD.EXE.

Управление памятью

- Память является важнейшим ресурсом, требующим тщательного управления со стороны ОС. Распределению подлежит вся оперативная память, не занятая ОС. Функциями ОС по управлению памятью являются отслеживание свободной и занятой памяти, выделение памяти процессам и освобождение памяти при завершении процессов, настройка адресов программы на конкретную область физической памяти и т.д. Все методы управления памятью могут быть разделены на два класса



(рис. 23).

Рис. 23. Классификация методов распределения памяти

- Распределение памяти фиксированными разделами.
- Самым простым способом управления оперативной памятью является разделение ее на несколько разделов фиксированной величины.
- Очередная задача, поступившая на выполнение, помещается либо в общую очередь (рис. 24, а), либо в очередь к некоторому разделу (рис. 24, б).

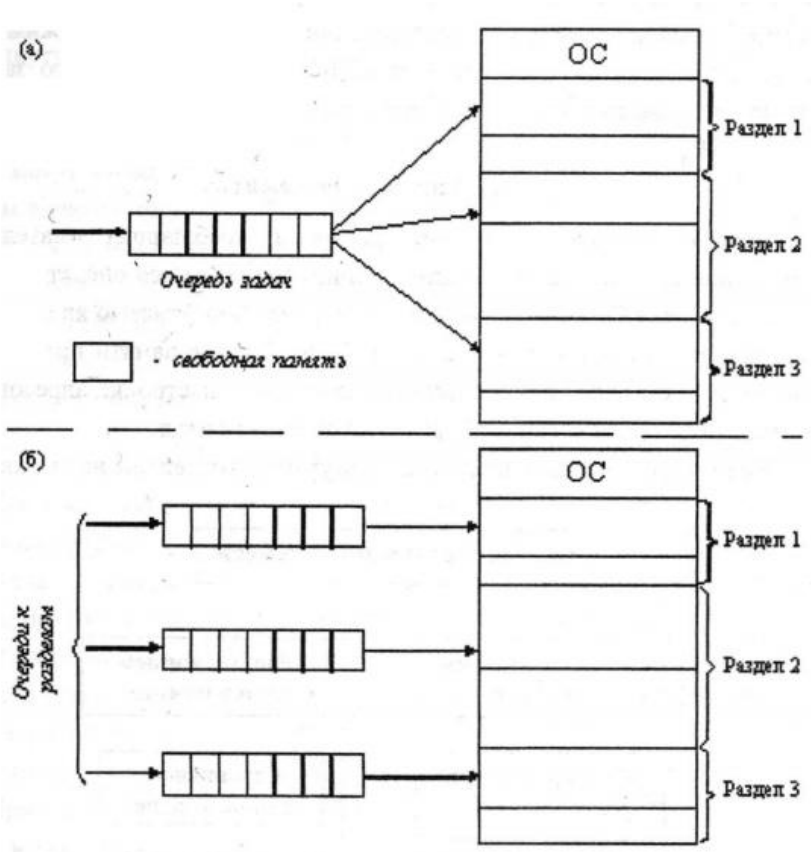


Рис. 24. Распределение памяти фиксированными разделами:
а - с общей очередью; б - с отдельными очередями

- 2. Распределение памяти динамическими разделами.

  Оперативная память ЭВМ не делится заранее на разделы. Сначала вся память свободна. Каждой вновь поступающей задаче выделяется необходимый ей объем оперативной памяти. Если достаточный объем оперативной памяти отсутствует, то задача не принимается на выполнение и стоит в очереди. После завершения задачи оперативная память освобождается, и на это место может быть загружена другая задача. Таким образом, в произвольный момент времени оперативная память представляет собой случайную последовательность занятых и свободных участков (разделов) произвольного размера.

- По сравнению с методом распределения памяти фиксированными разделами данный метод обладает гораздо большей гибкостью, но ему присущ очень серьезный недостаток - фрагментация памяти.

- 3. Распределение памяти перемещаемыми разделами.

- Одним из методов борьбы с фрагментацией является перемещение всех занятых участков в сторону старших либо в сторону младших адресов оперативной памяти, так, чтобы вся свободная оперативная память образовывала единую, свободную область.

- В дополнение к функциям, которые выполняет ОС при распределении оперативной памяти разделами переменной величины, в данном случае она должна еще время от времени копировать содержимое разделов из одного места оперативной памяти в другое, корректируя таблицы свободных и занятых областей. Эта процедура называется "сжатием". Хотя процедура сжатия и приводит к более эффективному использованию памяти, она может потребовать значительного времени, что часто перевешивает преимущества данного метода. Наиболее распространенными методами распределения памяти с использованием внешней памяти являются страничное, сегментное и странично-сегментное распределение памяти (вопросы организации этих моделей работы с памятью рассмотрены ранее).

Файловая система

- Файловая система - это часть операционной системы, назначение которой состоит в том, чтобы обеспечить пользователю удобный интерфейс при работе с данными, хранящимися на диске, и обеспечить совместное использование файлов несколькими пользователями и процессами. В широком смысле понятие "файловая система" включает:
- 1) совокупность всех файлов на диске,
- 2) наборы структур данных, используемых для управления файлами, таких, например, как каталоги файлов, дескрипторы файлов, таблицы распределения свободного и занятого пространства на диске,
- 3) комплекс системных программных средств, реализующих управление файлами, в частности: создание, уничтожение, чтение, запись, именование, поиск и другие операции над файлами.

- Файл — это абстрактный механизм, предоставляющий способ сохранять информацию на диске и считывать её и скрывающий от пользователя такие подробности, как способ, место хранения информации, детали работы дисков. Файлы идентифицируются именами. Пользователи присваивают файлам символьные имена, при этом учитываются определенные ограничения, накладываемые ОС. Различают обычные файлы, специальные файлы, файлы-каталоги.

- Логически файл представляет собой последовательность определенным образом организованных логических записей. Логическая запись – это наименьший элемент данных, которым можно оперировать при работе с файлом. Записи могут быть фиксированной длины или переменной длины. Они могут быть расположены в файле последовательно (последовательная организация), например, в файловых системах ОС UNIX и MS-DOS файл имеет простейшую логическую структуру - последовательность однобайтовых записей. Кроме того, записи могут быть расположены в файле в более сложном порядке, с использованием так называемых индексных таблиц, позволяющих обеспечить быстрый доступ к отдельной логической записи (индексно-последовательная организация).

- Физическая организация файла описывает правила расположения файла на устройстве внешней памяти. Файл состоит из физических записей - блоков. Блок - наименьшая единица данных, которой внешнее устройство обменивается с оперативной памятью. Выделяют 4 способа организации физического размещения файлов:

- 1) непрерывное размещение;
- 2) размещение в виде связанного списка блоков;
- 3) размещение с использованием связанного списка индексов;
- 4) перечень номеров блоков (рис.26).

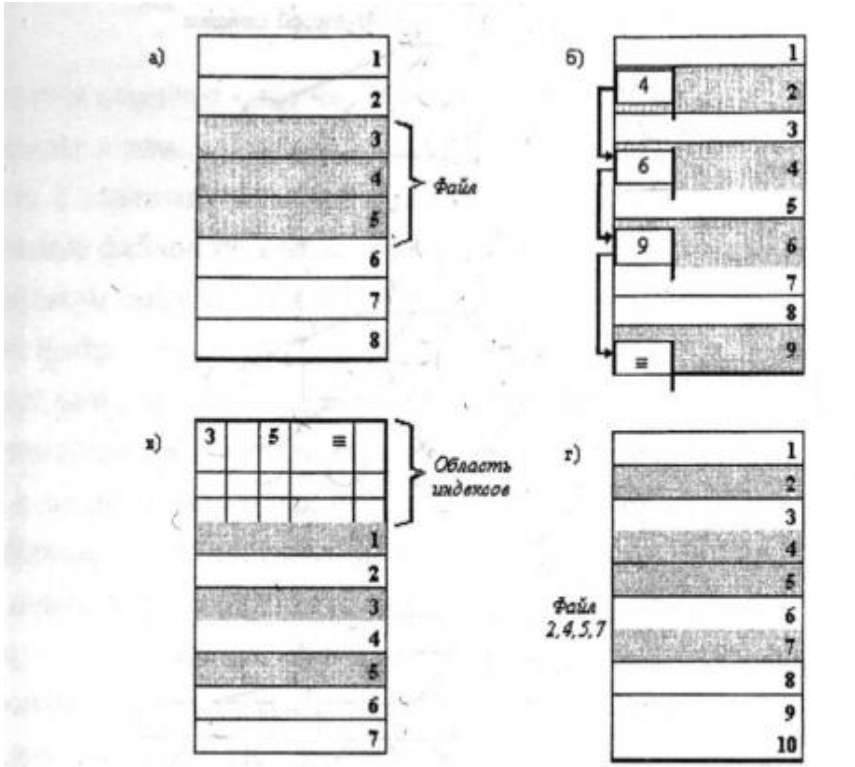


Рис. 26. Физическая организация файла
а - непрерывное размещение; б - связанный список блоков;
в - связанный список индексов; г - перечень номеров блоков

- Определить права доступа к файлу - значит определить для каждого пользователя набор операций, которые он может применить к данному файлу. В разных файловых системах может быть определен свой список операций доступа. Этот список может включать следующие операции: создание файла, уничтожение файла, открытие файла, закрытие файла, чтение файла, запись в файл, дополнение файла, поиск в файле, получение атрибутов файла, установление новых значений атрибутов, переименование, выполнение файла, чтение каталога и другие операции с файлами и каталогами.

- В самом общем случае права доступа могут быть описаны матрицей прав доступа, в которой столбцы соответствуют всем файлам системы, строки - всем пользователям, а на пересечении строк и столбцов указываются разрешенные операции. В некоторых системах пользователи могут быть разделены на отдельные категории. Для всех пользователей одной категории определяются единые права доступа. Например, в системе UNIX все пользователи подразделяются на три категории: владельца файла, членов его группы и всех остальных.

- Функционирование любой файловой системы можно представить многоуровневой моделью (рис. 27), в которой каждый уровень предоставляет некоторый интерфейс (набор функций) вышележащему уровню, а сам, в свою очередь, для выполнения своей работы использует интерфейс нижележащего уровня.

Рис. 27. Общая модель файловой системы

СПАСИБО ЗА ВНИМАНИЕ!