МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Кафедра «АСУ»

Конспект лекций

по дисциплине:

«Системное программное обеспечение»

студент гр. АСОИ-081

Дегтярев А. Н.

преподаватель

Зайченко Е.А.

Могилев, 2011

Вопросы к экзамену «Системное программное обеспечение»

1. Состав системного программного обеспечения. Операционная система (ОС) как посредник между уровнем пользователя и аппаратурой ЭВМ.
2. Классификация операционных систем.
3. Структура ОС: ядро и вспомогательные модули.
4. Концепция микроядерной архитектуры ОС.
5. Понятия вычислительного процесса и ресурса. Классификация ресурсов
6. Реализация понятия «последовательный процесс»
7. Основные виды ресурсов ВС и возможности их разделения.
8. Понятие прерывания и схема их обработки.
9. Типы прерываний. Механизм прерываний. Приоритеты прерываний
10. Мультипрограммирование, многопользовательский режим работы и режим разделения времени.
11. Диаграмма состояний процесса. Реализация понятия последовательного процесса в операционных системах
12. Основные виды ресурсов и возможности их разделения.
13. Понятия «процесс», «поток», «задание».
14. Создание процессов и потоков. Контекст и дескриптор.
15. Планирование и диспетчеризация потоков.
16. Состояния потока.
17. Алгоритмы планирования и диспетчеризации.
18. Средства взаимодействия процессов и потоков
19. Цели и средства синхронизации процессов и потоков. Гонки.
20. Критическая секция. Блокирующие переменные. Семафоры.
21. Понятие тупиковой ситуации при выполнении параллельных вычислительных процессов
22. Методы борьбы с тупиками.
23. Архитектура ОС Windows.
24. Процессы и потоки ОС Windows. Алгоритмы их планирования.
25. Состояния процессов и потоков ОС Windows.
26. Функции ОС по управлению памятью. Типы адресов.
27. Алгоритмы распределения памяти. Свопинг и виртуальная память.
28. Распределение памяти фиксированными, динамическими и перемещаемыми разделами.
29. Страничная организация виртуальной памяти.
30. Сегментная организация виртуальной памяти.
31. Сегментно-страничная организация виртуальной памяти.. Разделяемые сегменты памяти.
32. Архитектура памяти ОС Windows.
33. Реализация страничного распределения в ОС Windows.
34. Размещение объектов в памяти. Куча и стек.
35. Принцип действия кэш-памяти. Проблема согласования данных.
36. Способы отображения основной памяти на кэш.
37. Управления вводом-выводом.
38. Функции подсистемы ввода-вывода.
39. Многослойная модель подсистемы ввода-вывода.
40. Цели и задачи файловой системы.
41. Типы файлов. Иерархическая структура файловой системы. Имена файлов. Атрибуты файлов.
42. Физическая организация файловой системы.
43. Способы физической организации файла.
44. Файловая система FAT.
45. Файловая система NTFS.
46. Контроль доступа к файлам.
47. Физическая организация ФС UNIX.

1. Состав системного программного обеспечения. Операционная система (ОС) как посредник между уровнем пользователя и аппаратурой ЭВМ.

Структура вычислительной системы.

Операционная система (ОС) – это программа, которая обеспечивает возможность рационального использования оборудования компьютера удобным для пользователя образом.

В соответствии с этим определением ОС выполняет две группы функций:

- предоставление пользователю или программисту вместо реальной аппаратуры компьютера расширенной виртуальной машины, с которой удобней работать и которую легче программировать;
- повышение эффективности использования компьютера путем рационального управления его ресурсами в соответствии с некоторым критерием.

Управление ресурсами включает решение следующих общих задач, не зависящих от типа ресурса:

- планирование ресурса – то есть определение, какому процессу, когда и в каком количестве (если ресурс может выделяться частями) следует выделить данный ресурс;
- удовлетворение запросов на ресурсы;
- отслеживание состояния и учет использования ресурса – то есть поддержание оперативной информации о том, занят или свободен ресурс и какая доля ресурса уже распределена;
- разрешение конфликтов между процессами.

Основные функции ОС

- Планирование заданий и использования процессора.
- Обеспечение программ средствами коммуникации и синхронизации.
- Управление памятью.
- Управление файловой системой.
- Управление вводом-выводом.
- Обеспечение безопасности

Краткая история эволюции вычислительных систем

- Первый период (1945–1955 гг.). Ламповые машины. Операционных систем нет
- Второй период (1955 г.–начало 60-х). Компьютеры на основе транзисторов. Пакетные операционные системы
- Третий период (начало 60-х – 1980 г.). Компьютеры на основе интегральных микросхем. Первые многозадачные ОС
- Четвертый период (с 1980 г. по настоящее время). Персональные компьютеры. Классические, сетевые и распределенные системы

Наиболее характерными критериями эффективности вычислительных систем являются:

- пропускная способность – количество задач, выполняемых вычислительной системой в единицу времени;
- удобство работы пользователей, заключающееся, в частности, в том, что они имеют возможность интерактивно работать одновременно с несколькими приложениями на одной машине;
- реактивность системы – способность системы выдерживать заранее заданные (возможно, очень короткие) интервалы времени между запуском программы и получением результата.

2. Классификация операционных систем.

1. Назначение (универсальные, специализированные – управление производством, обучение)
2. Способ загрузки (загружаемые, постоянно находящиеся в памяти)
3. Особенности алгоритмов управления ресурсами
   3.1. Многозадачность:
   - однозадачные (MS DOS),
   - невытесняющая многозадачность (Windows 3.x, NewWare),
   - вытесняющая многозадачность (Windows NT, OS/2, Unix)

При невытесняющей многозадачности момент приостановки одной задачи и время на выполнение другой задачи определяется разработчиков программы, а при вытесняющей – решение о переключении процессора с выполнения одной задачи на другую принимается ОС.

3.2. Многопользовательский режим:
   - отсутствие (MS DOS, Windows 3.x),
   - имеется (Windows NT, OS/2, Unix)

Многопользовательская система обладает средствами защиты информации каждого пользователя от несанкционированного доступа других пользователей.

3.3. Многопроцессорная обработка:
отсутствие, асимметричные ОС, симметричные ОС.
Ассиметричные ОС выполняются на одном процессоре системы, распределяя остальные процессоры на прикладные задачи, симметричные ОС – децентрализованные.

4. По базовой технологии (Юникс-подобные или подобные Windows)
5. По типу лицензии (проприетарная или открытая)
6. Область использования и форма эксплуатации
   - пакетная обработка (OS/360)
   - разделение времени
   - реальное время (QNX)

Пакетная обработка: критерий эффективности – максимальная пропускная способность, формируется мультипрограммная смесь/набор задач для одновременного выполнения, предъявляют отличающиеся требования к ресурсам. Выбор нового задания зависит от текущей ситуации, выбирается выгодное системное задание.
Системное разделение времени: каждому пользователю или процессу выделяется квант процессорного времени, поскольку квант небольшой, у одновременно работающих пользователей складывается впечатление единоличной работы. Критерий эффективности – удобство пользователя.
Системы реального времени: применяются для управления техническими объектами; подразделяются на системы жесткого реального времени и мягкого реального времени. Для систем мягкого реального времени допустима одна сотая процента отказов с возможностью повторного выполнения операции (система бронирования билетов и т.д.)
Системы жесткого реального времени – отказы недопустимы, программное обеспечение представляет собой фиксированный набор программ, а выбор программы на выполнение

осуществляется в соответствии с расписанием. Для таких систем характерно дублирование ресурсов.

7. Аппаратная платформа
   7.1. ОС для смарт-карт (с интерпретатором виртуальной Java-машины)
   7.2. Встроенные ОС (Palm OS, Windows CE –Consumer Electronics)
   7.3. ОС для ПК (Windows 9.x, Windows 2000, Linux, Mac OS X)
   7.4. ОС мини-ЭВМ (UNIX для PDP-7)
   7.5. ОС мэйнфреймов (OS/390 – пакетная обработка, разделение времени, обработка транзакций)
   7.6. Серверные операционные системы для ЛВС, Интранет и Интернет (UNIX, AIX, Windows 2000/2002, Linux)
   7.7. Кластерные операционные системы (Windows 2000 Cluster Server, Sun Cluster (Solaris))

3. Структура ОС: ядро и вспомогательные модули.

Наиболее общим подходом к структуризации операционной системы является разделение всех ее модулей на две группы:
- ядро – модули ОС, выполняющие основные функции;
- модули, выполняющие вспомогательные функции ОС.

Состав ядра:
- Модули ядра выполняют такие базовые функции ОС, как управление процессами, памятью, устройствами ввода-вывода.
- В состав ядра входят функции, решающие внутрисистемные задачи организации вычислительного процесса, такие, как переключение контекстов, загрузка/выгрузка страниц, обработка прерываний. Эти функции недоступны для приложений.

Другой класс функций ядра служит для поддержки приложений, создавая для них так называемую прикладную программную среду.

Приложения могут обращаться к ядру с запросами – системными вызовами – для выполнения тех или иных действий, например, для открытия и чтения файла, вывода графической информации и т. д

Функции ядра, которые могут вызываться приложениями, образуют интерфейс прикладного программирования – API.

Вспомогательные модули ОС обычно подразделяются на следующие группы:
- утилиты – программы, решающие отдельные задачи управления и сопровождения компьютерной системы, такие, как программы архивирования данных;
- системные обрабатывающие программы – текстовые или графические редакторы, компиляторы, отладчики;
- программы предоставления пользователю дополнительных услуг – специальный вариант пользовательского интерфейса, калькулятор;
- библиотеки процедур различного назначения, упрощающие разработку приложений, например библиотека математических функций и т. д.

Для обеспечения высокой скорости работы ОС все модули ядра или большая их часть постоянно находятся в оперативной памяти, то есть являются резидентными.

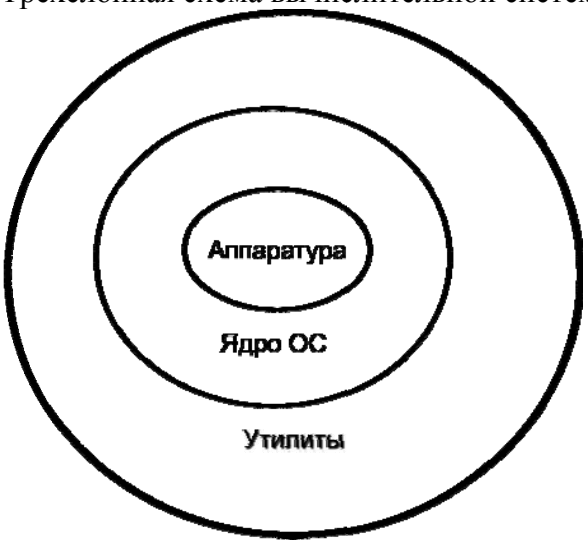
Модули ОС, оформленные в виде утилит, системных обрабатывающих программ и библиотек, обычно загружаются в оперативную память только на время выполнения своих функций, то есть являются транзитными.

Ядро и привилегированный режим

Для надежного управления ходом выполнения приложений операционная система должна иметь по отношению к приложениям определенные привилегии.

Аппаратура компьютера должна поддерживать как минимум два режима работы – пользовательский режим (user mode) и привилегированный режим, который также называют режимом ядра (kernel mode) или режимом супервизора (supervisor mode).
Приложения ставятся в подчиненное положение за счет запрета выполнения в пользовательском режиме некоторых критичных команд, связанных с переключением процессора с задачи на задачу, управлением устройствами ввода-вывода, доступом к механизмам распределения и защиты памяти.
Выполнение некоторых инструкций в пользовательском режиме запрещается безусловно (очевидно, что к таким инструкциям относится инструкция перехода в привилегированный режим), тогда как другие запрещается выполнять только при определенных условиях.
Полный контроль ОС над доступом к памяти достигается за счет того, что инструкции конфигурирования механизмов защиты памяти разрешается выполнять только в привилегированном режиме.
Системный вызов привилегированного ядра инициирует переключение процессора из пользовательского режима в привилегированный, а при возврате к приложению – переключение из привилегированного режима в пользовательский.
Во всех типах процессоров из-за дополнительной двукратной задержки переключения переход на процедуру со сменой режима выполняется медленнее, чем вызов процедуры без смены режима.
Архитектура ОС, основанная на привилегированном ядре и приложениях пользовательского режима, считается классической.

Многослойная структура ОС
Трехслойная схема вычислительной системы



Ядро может состоять из следующих слоев:
- Средства аппаратной поддержки
- Машинно-зависимые компоненты ядра
- Базовые механизмы ядра
- Менеджеры ресурсов
- Интерфейс системных вызовов

Средства аппаратной поддержки ОС
До сих пор об операционной системе говорилось как о комплексе программ, но часть функций ОС может выполняться и аппаратными средствами.
Типичный набор средств аппаратной поддержки ОС:
- средства поддержки привилегированного режима;
- средства трансляции адресов;
- средства переключения процессов;

- система прерываний;
- системный таймер;
- средства защиты областей памяти.

1. Базовые механизмы ядра

Этот слой выполняет наиболее примитивные операции ядра, такие, как программное переключение контекстов процессов, диспетчеризацию прерываний, перемещение страниц из памяти на диск и обратно и т. п. Модули данного слоя не принимают решений о распределении ресурсов – они только отрабатывают принятые «наверху» решения, что и дает повод называть их исполнительными механизмами для модулей верхних слоев.

2. Менеджеры ресурсов

Этот слой состоит из мощных функциональных модулей, реализующих стратегические задачи по управлению основными ресурсами вычислительной системы. Обычно на данном слое работают менеджеры процессов, ввода-вывода, файловой системы и оперативной памяти.

Каждый из менеджеров ведет учет свободных и используемых ресурсов определенного типа и планирует их распределение в соответствии с запросами приложений.

3. Интерфейс системных вызовов

Этот слой является самым верхним слоем ядра и взаимодействует непосредственно с приложениями и системными утилитами, образуя прикладной программный интерфейс операционной системы.

Функции API, обслуживающие системные вызовы, предоставляют доступ к ресурсам системы в удобной и компактной форме, без указания деталей их физического расположения.

4. Концепция микроядерной архитектуры ОС.

В привилегированном режиме остается работать только очень небольшая часть ОС, называемая микроядром

Микроядро защищено от остальных частей ОС и приложений.

В состав микроядра обычно входят машинно-зависимые модули, а также модули, выполняющие базовые функции ядра по управлению процессами, обработке прерываний, управлению виртуальной памятью, пересылке сообщений и управлению устройствами ввода-вывода, связанные с загрузкой или чтением регистров устройств.

Все остальные более высокоуровневые функции ядра оформляются в виде приложений, работающих в пользовательском режиме.

Менеджеры ресурсов, являющиеся неотъемлемыми частями обычного ядра – файловая система, подсистемы управления виртуальной памятью и процессами, менеджер безопасности, – становятся «периферийными» модулями, работающими в пользовательском режиме и называются серверами ОС

Механизм обращения к функциям ОС, оформленным в виде серверов

- Каждое приложение пользовательского режима работает в собственном адресном пространстве и защищено тем самым от вмешательства других приложений. Непосредственная передача сообщений между приложениями невозможна.
- Клиент, которым может быть либо прикладная программа, либо другой компонент ОС, запрашивает выполнение некоторой функции у соответствующего сервера, посылая ему сообщение.
- Микроядро, имеет доступ к адресным пространствам каждого из этих приложений и поэтому может работать в качестве посредника.
- Микроядро сначала передает сообщение, содержащее имя и параметры вызываемой процедуры нужному серверу, затем сервер выполняет запрошенную операцию, после чего микроядро возвращает результаты клиенту с помощью другого сообщения.

Таким образом, работа микроядерной операционной системы соответствует известной модели клиент-сервер, в которой роль транспортных средств выполняет микроядро.
Достоинства микроядерной архитектуры: единообразные интерфейсы, расширяемость, переносимость, гибкость, надежность, поддержка распределенных систем, поддержка объектно-ориентированных ОС.
Недостатки микроядерной архитектуры: при классической организации ОС выполнение системного вызова сопровождается двумя переключениями режимов, а при микроядерной организации – четырьмя. Таким образом, операционная система на основе микроядра при прочих равных условиях всегда будет менее производительной, чем ОС с классическим ядром.
Монолитные системы.
Для построения монолитной системы необходимо скомпилировать все отдельные процедуры, а затем связать их вместе в единый объектный файл с помощью компоновщика (примерами могут служить ранние версии ядра UNIX или Novell NetWare). Каждая процедура видит любую другую процедуру (в отличие от структуры, содержащей модули, в которой большая часть информации является локальной для модуля, и процедуры модуля можно вызвать только через специально определенные точки входа).

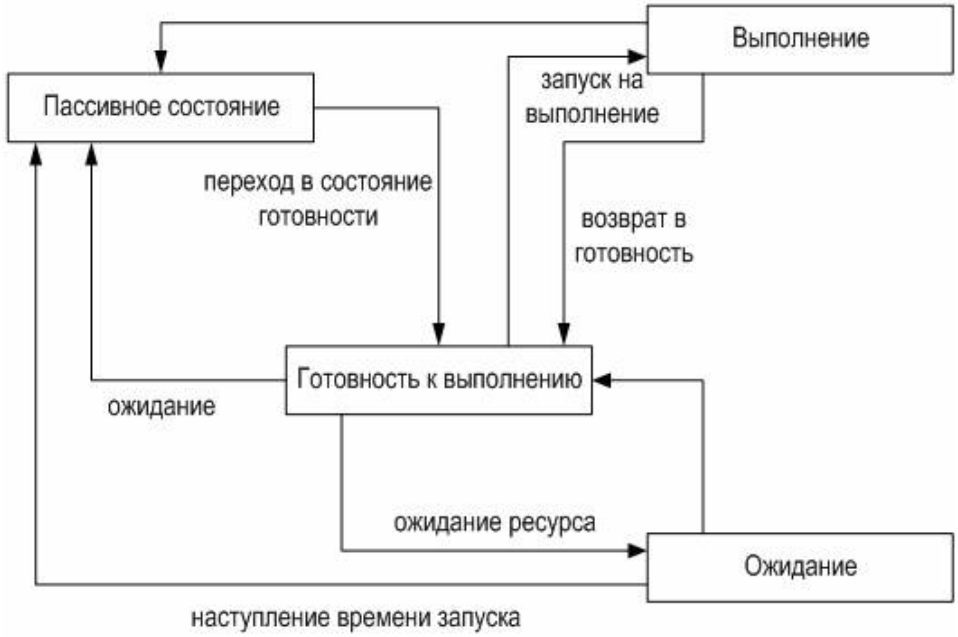
5. Понятия вычислительного процесса и ресурса. Классификация ресурсов

Вычислительный процесс − это процесс выполнения программы совместно с ее данными на процессоре (редактирование текста, трансляция, выполнение какой-либо программы).
Под ресурсом понимают некоторый объект, который обладает свойствами повторного и неоднократного использования процессами, запрашивающими, использующими и освобождающими ресурсы.
Ресурс – это абстрактная структура с некоторым набором атрибутов, которые характеризуют ее функциональные характеристики и способы доступа к ней.
Концепция ресурса определяется целью выработать механизмы распределения и управления ресурсами.
На начальном этапе ресурсами считались: процессорное время, память, методы ввода/вывода, периферийные устройства, в современных системах к ним добавились программные и информационные ресурсы, такие как сообщения, файлы и т.д..
Классификация ресурсов:
Ресурсы:
1. Делимые
    1.1. Одновременно разделяемые (жесткий диск)
    1.2. Параллельно разделяемые (процессорное время, данные)
2. Неделимые (принтер)

Для того, чтобы ОС могла выделять ресурсы и контролировать их исполнение, она должна поддерживать информационные структуры, называемые управляющими таблицами.
При мультипрограммировании повышается загрузка ресурсов и увеличивается время выполнения каждой задачи, за счет затрат времени на ожидание, освобождение ресурса. В многозадачной системе ресурс может быть выделен в задаче, если:
1. Ресурс свободен и на него нет запросов от задачи с более высоким приоритетом.
2. Текущий и предыдущий запросы позволяют использовать ресурс совместно, тогда ресурс делится.
3. Ресурс используется задачей более низкого приоритета, то он временно отбирается.

Если одно из перечисленных условий выполнено, то задача выполняется с выделенным ресурсом, в противном случае задача ставится в очередь к ресурсом и переводится в режим ожидания.
Управление ресурсами организуется на основе некоторой стратегии или набора правил, обеспечивающих полную эффективность использования ресурсов.

Состояние процесса.
Все управляющие процессы подразделяются на 3 вида:
1. Управляющие процессы ядра ОС, занимающиеся распределением ресурсов.
2. Системные обрабатывающие процессы, ресурсы для них назначаются по определенному заранее правилу.
3. Процессы пользователя. Процесс может находиться в пассивном и активном состояниях. Он присутствует в системе, но ему не требуются ресурсы. В активном состоянии процесс участвует в конкуренции за ресурсы. Кроме того он может находится в следующих подсостояниях:
- выполнение – обладание всеми необходимыми ресурсами и их использование для решения задачи.
- готовность к выполнению – наличие всех ресурсов, кроме процессорного времени.
- ожидание – ресурсы не могут быть предоставлены процессу из-за их занятости другими процессами.
За время существования процесс неоднократно переходит из одного состояния в другое. Переход из состояния в состояние определяется: затратами ресурсов, системными функциями и другими процессами.



Переход в состояние готовности из пассивного состояния возможен:
1. По команде пользователя.
2. При выборе процесса планировщиком из очереди (исполнение пакетных или командных файлов).
3. По вызову другой задачи посредством ядра ОС.
4. По прерыванию из внешнего устройства.
5. По наступлению запланированного времени запуска программы.
Из подсостояния выполнения процесс может выйти по следующим причинам:
1. Завершение (ОС может перевести процесс в пассивное состояние до следующего вызова или уничтожить его)
2. Переход в состояние готовности в связи с появлением более приоритетной задачи или окончания кванта времени.
3. Блокировки из-за невозможности предоставления ресурсов либо ввода/вывода информации, либо по команде пользователя.

6. Реализация понятия «последовательный процесс»

Для отслеживания состояний и управления процессами ОС создает для каждого процесса специальную информационную структуру. В разных ОС эта структура может носить названия: - «описатель задачи», блок управления задачи. Современное принятое название такой структуры – дескриптор процесса.
Дескриптор процесса – информационная структура, содержащая:
1. Идентификатор процесса
2. Тип процесса, определяющий правила предоставления ресурсов.
3. Приоритет процесса.
4. Переменную состояния (значение этой переменной зависит от того состояния, в котором находится процесс)
5. Защищенную область памяти с контекстом задач (это текущее значение ресурсов процессора на момент прерывания работы процессора).
6. Информация о ресурсах (указания на открытые файлы).
7. Место (область памяти/общения с другими процессами).
8. Параметры времени запуска (момент времени активации, периодичность запуска).
Дескриптор располагается в ОП. Операционная система формирует из них списки и отображает изменение состояния процесса перемещение его дескриптора из одного списка в другой. Список еще называют очередями.
В некоторых ОС количество дескрипторов определено жестко (системы реального времени), в других может изменяться (windows). Средствами аппаратной поддержки многозадачности в процессорах семейства Intel являются регистры TR.
Для увеличения показателей мультипрограммирования в современной ОС вводится понятие потока. Процесс может содержать один или несколько потоков. Они находятся внутри процесса и не имеют собственных ресурсов, каждому из них выделяются лишь собственный квант процессорного времени. Дескриптор потока строится на основе дескрипторов процессов.

7. Основные виды ресурсов и возможности их разделения.

1. Центральный процессор (ЦП). Каждому процессу выделяется квант времени.
2. Память. Может делиться параллельно между несколькими процессами, которые одновременно в ней существуют, либо последовательно по очереди в порядке выполнения. Параллельное деление более эффективно в плане загрузки процессора, но ухудшает условия выполнения для каждого процесса в отдельности, применяется совместно с системной областью на диске (механизм виртуальной памяти)
3. Внешняя память. Для внешней памяти ресурсов считается как ее объем, так и доступ к ней. Память может выделяться одновременно, а доступ к ней – попеременно. Часть устройств внешней памяти (диски) разделяются между процессами параллельно, так как используют механизм прямого доступа. Устройства последовательного доступа (стримеры), являются неделимым ресурсом.
4. Программные модули. Они бывают однократно используемыми (неделимые ресурсы) – модули загрузки ОС, и многократно используемыми (разделяемые ресурсы) – утилиты, подпрограммы. Многократно используемые модули делятся на:
- Привилегированные, порядок которых не могут нарушить внешние события. Они являются попеременно разделяемым ресурсом.
- Непривилегированные, работают в обычном режиме, включающем систему прерывания, не являются разделяемыми ресурсами.
- Реентерабельные модули, допускают повторное обращение или повторные прерывания, являются разделяемыми ресурсами.

- Повторно-входимые модули, допускают повторное обращение, но не нельзя прерывать, являются попеременно-разделяемыми ресурсами.
5. Информационные ресурсы (данные). Существуют в виде переменных в ОП или файлов на диске. Разделяемыми ресурсами являются данные, используемые процессом только для чтения. Если процессы могут изменять информационные ресурсы, то ОС должна организовать работу с такими данными.

8. Понятие прерывания и схема их обработки.

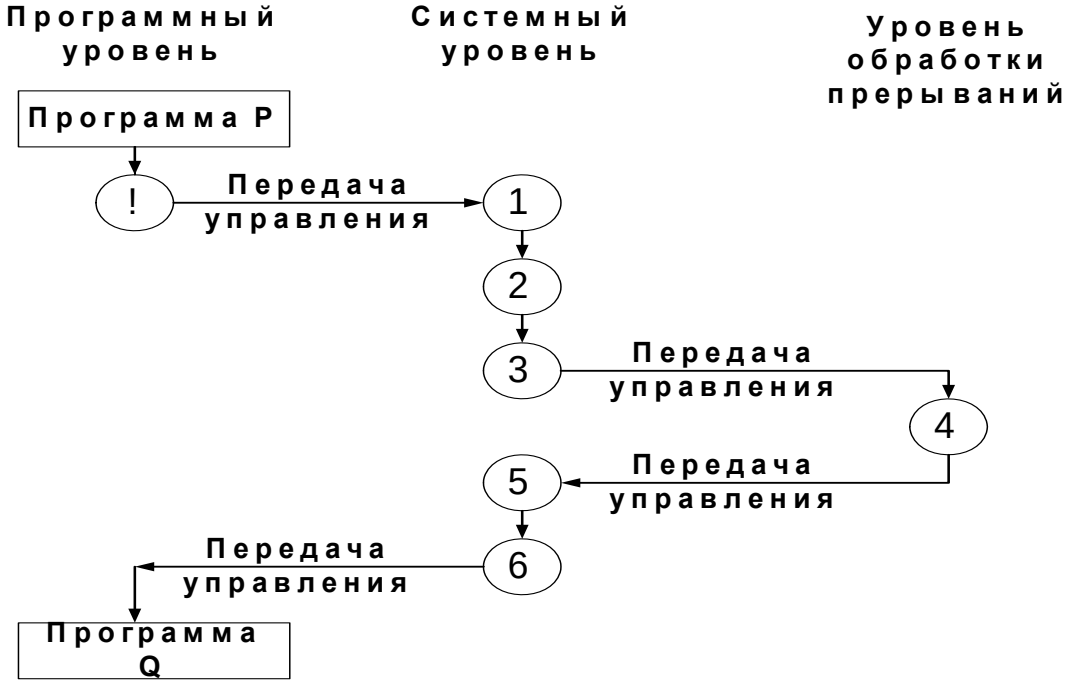
Прерывание – принудительная передача управления от выполняемой программы к системе, происходящая при возникновении определенного события.

Механизм прерывания позволяет установить параллельное функционирование отдельных устройств вычислительной системы и реагировать на особые состояния, возникающие при работе процессора.

Главные функции механизмов прерывания:

1. Распознание и классификация прерывания.
2. Передача управления обработчику прерывания (специальная системная программа – обработчик).
3. Корректное возвращение к прерванной программе.

Схема обработки прерываний.



Обозначения:

! – прерывание (сигнал – установление факта прерывания)

1 – идентификация прерывания

2 – отключение других прерываний

3 – смена контекста-1 (сохраняет состояния прерванного процесса из системных регистров)

4 – обработка прерывания, включающая определение программы Q, которую следует запустить.

5 – смена контекста-2

6 – установка прежнего режима системы прерываний. Передача управления осуществляется аппаратно, программа Q может быть прервана программой P.

9. Типы прерываний. Механизм прерываний. Приоритеты прерываний

В зависимости от источника прерываний они делятся на:
1. Внешние (аппаратные прерывания): возникают в результате действий пользователя или в результате поступления сигналов от аппаратных устройств. Этот план является асинхронным по отношению к потоку инструкций прерывания программы.
2. Внутренние (исключения): происходит синхронно выполнение программы при появлении аварийной ситуации в ходе исполнения некоторых инструкций программы (деление на 0, ошибки точности, переполнение, обращение к запрещенной области памяти)
3. Программные: происходят при выполнении особой команды процессора, выполнение которой организует переход на новую последовательность инструкций, а также переход в привилегированный режим.

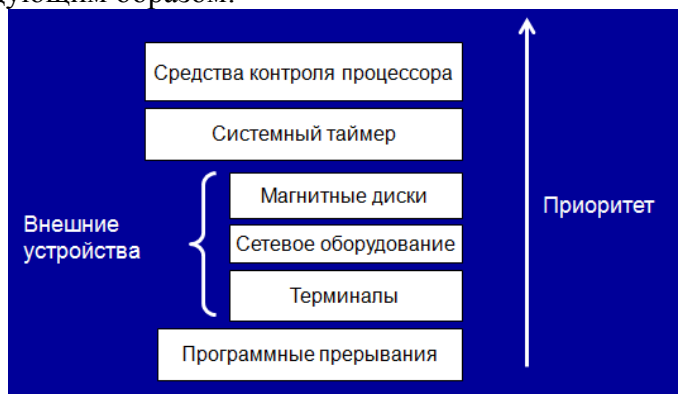
Механизм прерывания поддерживается как аппаратными средствами, так и программными средствами ОС. Существует 2 основных способа, с помощью которых выполняются прерывания:
1. Векторный.
2. Опрашиваемый.

Прерываниям присваивается приоритет с помощью которого они ранжируются по важности. Если прерывания имеют одинаковый приоритет, то они относятся к одному уровню прерываний.

Векторные прерывания: устройством, использующим этот способ, назначается вектор прерывания, представляющий собой сигнал, несущий информацию о закрепленным за данным устройством номере и адрес программы обработчика прерывания.

Опрашиваемый способ: процессор получает от устройства только информацию об уровне приоритета прерывания. В каждом уровне не может быть связано несколько устройств и несколько обработчиков. Чтобы определить какое именно устройство запросило прерывание – вызываются все обработчики этого уровня пока один из них не подтвердит, что прерывание произошло от его устройства. Могут использоваться одновременно 2 способа. Для того, чтобы не образовывались вложенные очереди прерывания, исполняются механизм маскирования, заключающийся в блокировке или отключении определенных уровней прерывания.

Приоритеты позволяют определить важность прерывания и в современных ОС распределяются следующим образом:



Приоритеты могут обслуживаться как относительные и абсолютные. Обслуживание запросов прерываний по схеме с относительным приоритетом заключается в том, что при одновременном поступлении запросов выбирается запрос, имеющий высший приоритет. Однако в дальнейшем процессора обработки прерываний не откладывается даже в том случае, когда появляется более приоритетные запросы. Решение о выборе нового запроса принимается только в момент завершения обслуживания текущего прерывания (относительного прерывания).

Если процессор работает по схеме с абсолютным приоритетом, то он поддерживает в одном из внутренних регистров переменную, фиксирующую уровень приоритета, обслуживающуюся в данный момент прерывания. При поступлении запроса из определенного класса его приоритет сравнивается с текущим приоритетом процессора и если приоритет запроса выше, то текущая процедура обработки прерывания вытесняется, а по завершении обслуживания нового прерывания происходит возврат к прерванной процедуре.
Программные прерывания выглядят для прикладного программиста как набор функций, с помощью которых можно упростить прикладную программу или выполнить действия, запрещенные в пользовательском режиме (например, обмен данными с устройством ввода/вывода).
В большинстве ОС системные вызовы обслуживаются по централизованной схеме; для обработки всех видов прерываний исполняет компонент ОС называемый диспетчером прерываний.
Диспетчер прерываний представляет собой программу, которая сохраняет содержимое регистра в системном стеке, проверяет попадает ли запрашиваемый номер в поддерживаемый диапазон и передает управление обработчику.

10. Мультипрограммирование, многопользовательский режим работы и режим разделения времени.
11. Диаграмма состояний процесса. Реализация понятия последовательного процесса в операционных системах

13. Понятия «процесс», «поток», «задание».

Для ОС единицей работы, претендентом на ресурсы является процесс.
Под объектом-процессом понимают программу и необходимые для ее выполнения ресурсы. Процесс может быть разбит на несколько параллельно выполняемых частей – потоки. Цели создания потоков – возможность увеличить производительность системы за счет распределения потоков по разным процессорам. Однако в современных ОС именно поток является исполнителем кода и при создании процесса для него автоматически создается хотя бы один поток.
Наименьшей единицей являются волокна (Fibers) – доступны в windows.
Несколько процессов могут быть объединены в задания.
Задание – набор процессов с общими квотами и инструментами.
Процесс – контейнер для ресурсов и потоков.
Поток – исполнитель кода в процессе.
Волокно – облегченный поток, полностью управляемый в пространстве пользователя.

14. Создание процессов и потоков. Контекст и дескриптор.

Создать процесс означает создать описатель процесса, в качестве которого выступает одна или несколько информационных структур, содержащих все сведения о процессе, необходимые для управления им.
Примером таких описателей являются:
1. Блок управления задачей в ОС (ОС-360).
2. Управляющий блок процесса (ОС/2).
3. Объект-процесс в Windows.
4. Дескриптор процесса в ОС Unix.
Создание процесса включает также загрузку кодов и данных с диска в оперативную память, в системную виртуальную память в начальный момент загружается только часть кодов и данных, а остальная часть копируется в область подкачки (специальная область

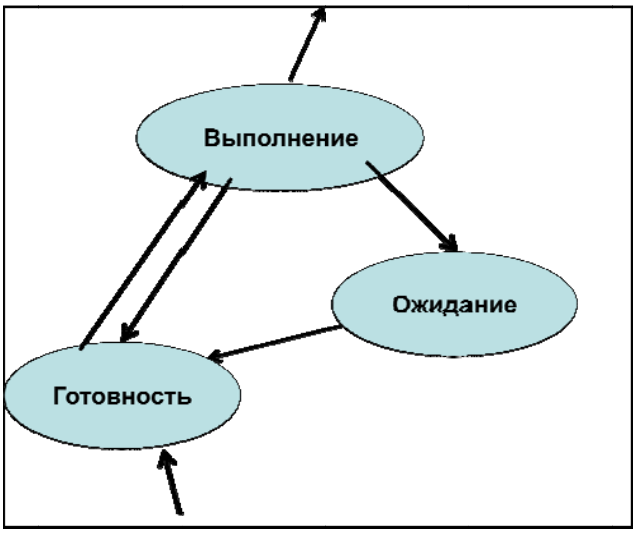
диска). При выполнении этих действий тесно взаимодействуют системы управления процессом и подсистема управления памятью и файловой системой.
В многопоточной системе при создании процесса генерируется как минимум один поток. Описатель потока – информационная структура, содержит идентификатор потока, данные о правах доступа и приоритете, о состоянии потока.
Во многих системах поток может обращаться к ОС с запросом на создание так называемых потоков-потомков. В разных ОС отношения между потомками и родителями отличаются, например: после выполнения родительского потока и его завершение синхронизируется с завершением всех его потомков, в других системах потомки работают асинхронно по отношению к родительскому потоку. Как правило, потоки наследуют многие свойства родительских потоков.
В качестве классического применения создания процесса рассмотрим процесс в ОС семейства Unix.
ОС использует два типа информационных структур: дескриптор и контекст процесса.
Дескриптор содержит информацию о процесса, необходимую ядру в течении всего жизненного процессорного цикла как в активном, так и в пассивном состояниях. Дескрипторы отдельных процессов объединяются в список, образуя таблицу процессов. Память для таблицы процессов отводится динамически в области ядра. На основании информации, содержащейся в таблице процессов осуществляется планирование и организация процессов.
В дескрипторе содержатся: идентификатор, сведения о приоритете, о состоянии, о расположении процесса в ОП и на диске, о событиях, которые ожидают процесс и пр.
Контекст содержит информацию о процессе, необходимую для возобновления выполняемого процесса с места его приостановки, содержит регистры, коды ошибок системных вызовов, открытых файлов и т.д.
Контекст характеризует состояние вычислительной среды в момент прерывания.

15. Планирование и диспетчеризация потоков.

На протяжении выполнения процесса его потоки могут быть неоднократно прерваны и продолжены. Работа по определению того, в какой момент прервать выполнение текущего активного потока и какому потоку предоставить выполнение, называется планированием.
ОС выполняет планирование потоков независимо от того принадлежат они одному или разным процессам. Принимается во внимание приоритет потоков, время ожидания в очереди и другие факторы.
Существует большое количество алгоритмов планирования. Все они могут быть разделены на 2 класса:
1. При динамическом (онлайн) планировании – решение принимается во время работы системы на основе анализа текущей ситуации. Этот способ применяется в системах общего назначения.
2. Статическое (оффлайн) планирование – применяется в системах реального времени. Весь набор задач определен заранее, сформирована таблица-расписание, в которой указывается какому процессу или потоку, когда и на какое время должен быть предоставлен процессор.
Диспетчеризация заключается в реализации найденного в результате планирования решения и включает в себя:
1. Сохранение контекста текущего потока, который требуется сменить
2. Загрузка контекста нового потока, выбранного в результате планирования.
3. Запуск нового потока на выполнение.
Переключение контекстов выполняется совместно с аппаратными средствами процессора.

16. Состояния потока.

Состояние потока



Для реализации планирования, ОС получает управление каждый раз, когда в системе происходит событие, требующее перераспределения процессорного времени. Такими событиями являются:

1. Произошло прерывание таймера сигнализирующего, что закончилось время, отведенное задаче на выполнение.
2. Активная задача выполнила системный вызов, связанный с запросом к ресурсу, который в настоящий момент занят, или на операции ввода/вывода.
3. Активная задача выполнила системный вызов, связанный с освобождением ресурсов. Планировщик проверяет, ожидает ли этот ресурс другая задача и если да, то эта задача переводится в состояние готовности.
4. Произошло аппаратное прерывание, сигнализирующее о завершении внешним устройством операций ввода/вывода.
5. Внутренние прерывания сигнализируют об ошибке, которая произошла в результате выполнения активной задачи.
6. Закончился код активной задачи.

а) поток или процесс обладает информационной структурой и требуемыми ресурсами.
б) планировщик выбрал поток на выполнение.

17. Алгоритмы планирования и диспетчеризации.

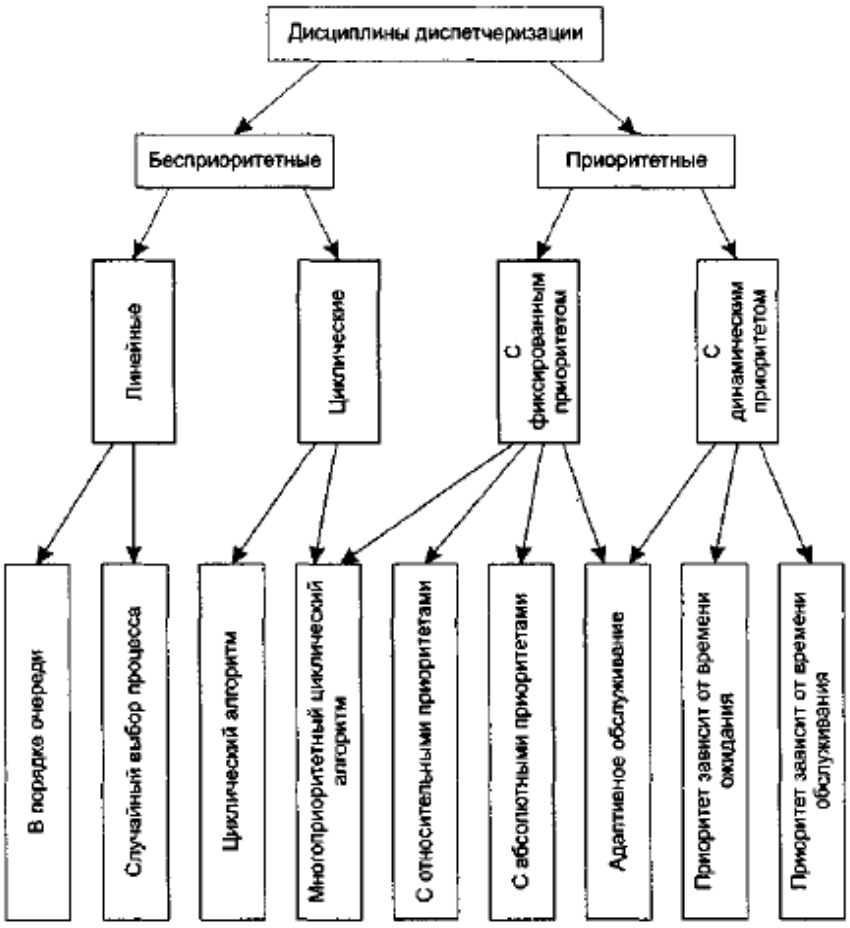
Все множество алгоритмов планирования можно разделить на 2 активных класса:

1. Вытесняющие алгоритмы (такие способы планирования потоков, в которых в решениях о переключениях процессора с выполнением одного потока на выполнение другого принимается ОС).
2. Невытесняющие алгоритмы (алгоритмы основаны на том, что активному потоку позволено выполняться пока он сам не передаст управление ОС, чтобы она выбрала из очереди другой готовый поток).

Достоинства вытесняющей многозадачности: разработчик пишет программу как если бы она выполнялась в однозадачном режиме; система полностью контролирует работу приложений и может приостанавливать их или снимать с выполнения в случае аварийных ситуаций.

Недостатки: выбранные ОС моменты времени для переключения контекстов потоков могут быть нерациональными (в момент работы с файлом, сохранения данных) и требовать значительных временных затрат. Это приводит к снижению производительности.

Достоинства невытесняющей многозадачности: разработчик расставляет систенмные вызовы (моменты приостановки) таким образом, чтобы обеспечить минимальные временные задержки.
Недостатки: управление приложением теряется до тех пор, пока само не передаст управление системе и, в случае краха приложения, его невозможно снять с выполнения.
Большинство систем общего назначения и систем реального времени используют вытесняющие алгоритмы. Невытесняющие алгоритмы применяются в файловых серверах, где за счет продуманному планированию достигается высокая скорость файловых операций.
Важным понятием в планировании является квант - непрерывный ограниченный период процессорного времени, предоставляемый потоку для выполнения. Поток, исчерпавший свой квант, переводится в состояние готовности.
Кванты, выделяемые потоку могут быть фиксированной величины или могут изменяться.
Если поток не отрабатывает свой квант из-за необходимости выполнять ввод/вывод данных, ему может быть предоставлена или не предоставлена «компенсация» при последующем выборе на выполнение.
В системах обычно существуют стандартно заданные величины квантов для пользовательских и системных процессов.
Приоритет – число, характеризующее степень привилегированности потока, при использовании вычислительных ресурсов.
В зависимости от использования приоритета выделяют следующие дисциплины диспетчеризации:



В системах с абсолютным приоритетом выполнение активного потока прерывается, если в очереди готовых потоков появился поток, приоритет которого выше активного потока.
Системы с относительным приоритетом: потоку позволено отработать свой квант времени, даже если в системе появился более приоритетный поток.

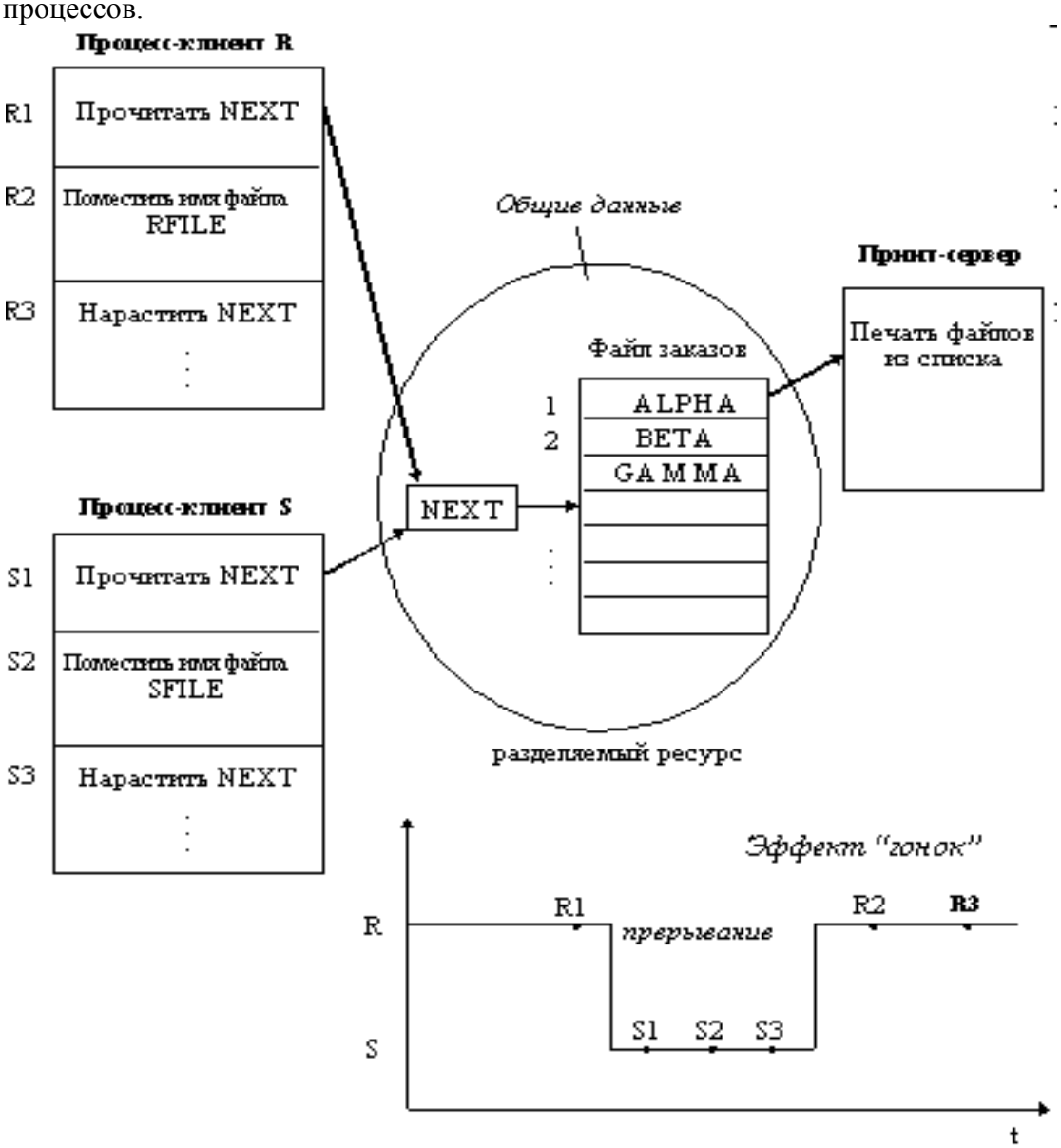
Независимо от использующегося алгоритма система должна гарантировать выполнение потока за некоторый период времени.
Большинство систем общего назначения используют смешанный алгоритм планирования, например: системные процессы, зарезервированные для ядра, используют стратегию фиксированных абсолютных приоритетов. Для пользовательских процессов применяется относительное приоритетное и динамическое адаптивное обслуживание.

18. Средства взаимодействия процессов и потоков

За время работы процессы неоднократно обращаются к ОС, а также к другим процессам. Выделяют следующие цели взаимодействия:
1. Передача данных.
2. Совместное использование информации (нескольким процессам необходимо обработать разделяемые данные таким образом, чтобы их изменение одним из них сразу же становилось видимым остальным процессам).
3. Уведомление о событиях (одному процессу требуется уведомлять процесс или набор процессов о возникновении какого-то события).
4. Совместное использование ресурсов (набор взаимодействий процессов иногда нуждается в определении собственного протокола доступа к ресурсам, поток строится поверх основного набора средств ОС).
5. Управление процессами (процесс, контролирующий управление другими процессами, может перехватывать все аппаратные и внутренние прерывания управляемых процессов и уведомляться обо всех изменениях их состояния).
Межпроцессорное взаимодействие – набор способов обмена данными между множеством потоков в одном или более процессах. Процессы могут быть запущены на одном или более компьютерах, связанных между собой сетью. IPC-способы делятся на методы обмена сообщениями, синхронизацией, разделяемой памяти и удаленных вызовов (RPC). Методы IPC зависят от пропускной способности и задержки взаимодействия между потоками и типом передачи данных.
IPC также может упоминаться как межпотоковое взаимодействия, межпоточное взаимодействия и межпрограммное взаимодействия.
Виды механизмов в IPC:
- Библиотека динамической компоновки (DLL): когда в рамках DLL объявляется переменная, ее можно сделать разделяемой, и все процессы, обращающиеся к библиотеке, для таких переменных будут использоваться одно и то же место в физической памяти.
- Сигналы: применяются для уведомления о некотором событии, произошедшем в системе.
- Анонимные каналы: полезны для организации прямой связи между двумя процессами на одном ПК.
- Именованные каналы: полезны для организации прямой связи между двумя процессами на одном ПК или в сети.
- Почтовые ячейки: полезны для организации связи одного процесса со многими на одном ПК или в сети.
- Гнезда: полезны для организации пересылки данных как в Windows-программы, так и в прочие программы, функционирующие на одном ПК, в сети или в итросети.
- Очереди сообщений Microsoft Message Queries (MSMQ): данный протокол может работать с удаленными процессами и даже с процессами, которые на данный момент недоступны (например, не запущены). Доставка сообщения по адресу гарантируется.
- Файлы отображаемой памяти: обеспечивают одновременный доступ к объектам файла-отображения из нескольких процессов.

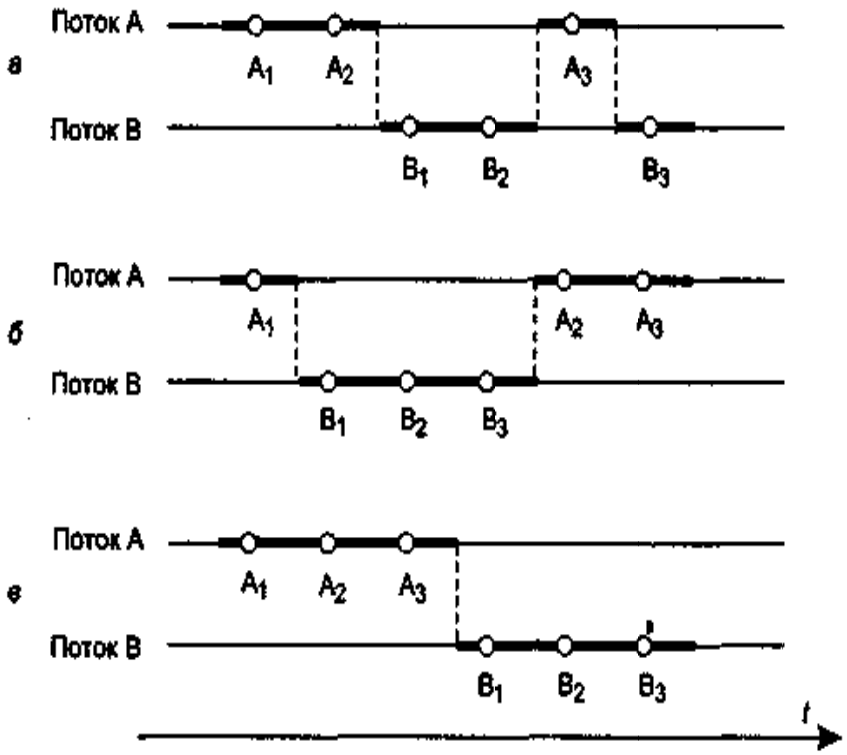
19. Цели и средства синхронизации процессов и потоков. Гонки.

В многозадачной системе моменты прерывания потоков, время нахождения его в очередях, разделением ресурсом, порядок выбора потока для его выполнения является результатом стечения многих обстоятельств и могут быть интерпретированы как случайные события.
Любое взаимодействие потоков или процессов связанное с синхронизацией, которая заключается в согласовании скорости их выполнения путем приостановки потока до наступления некоторого события и активизации потока при наступлении этого события.
Для синхронизации прикладных программ, программист может использовать как собственные средства, так и средства ОС.
Рассмотрим необходимость синхронизации на классическом примере взаимодействия процессов.



Ситуация, когда несколько процессов или потоков используют разделяемые ресурсы, события могут развиваться по одному из вариантов:
а, б) Возникает эффект гонок, приводящий к потери данных одного из процессов.
Результат выполнения процесса зависит от момента его приостановки и возобновления его работы.
Гонками называется ситуация, когда 2 и более процессов и поток обрабатывают разделяемые данные и конечный результат зависит от соотношения из скорости.

Сложность ситуации в том, что гонки возникают нерегулярно, большую часть времени процессы не мешают друг другу (в).



Для решения проблемы синхронизации необходимо проанализировать код процесса и выделить часть программы, в которой ведется работа с разделяемыми данными.

20. Критическая секция. Блокирующие переменные. Семафоры.

Критические данные – разделяемые данные при несогласованном изменении которых могу возникнуть нежелательные эффекты. В примере критическими данными является переменная Next.

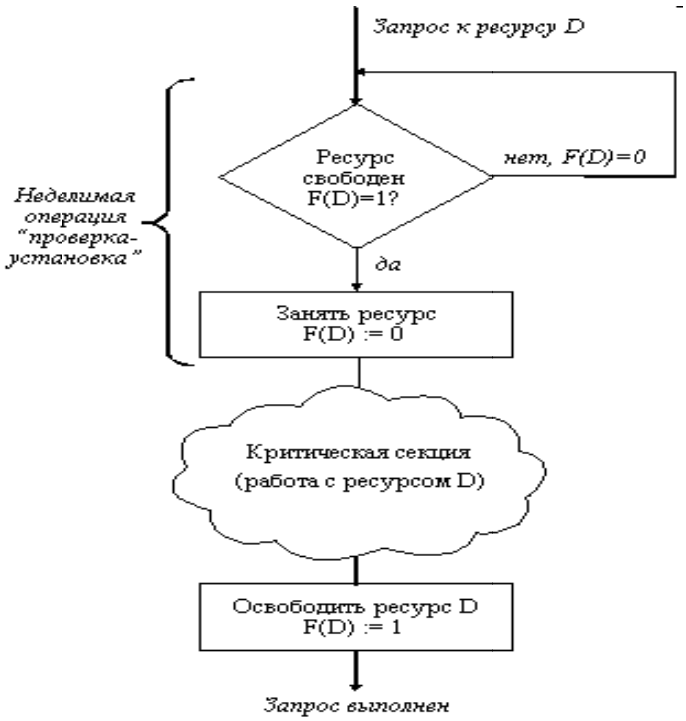
Критическая секция – это часть программы, результат выполнения которой может непредсказуемо меняться, если переменные, относящиеся к этой части программы изменяются другими процессами или потоками, когда выполнение этой части еще не завершено.

Чтобы исключить эффект гонок необходимо обеспечить принцип взаимного исключения: в каждый момент времени в критической секции должен находиться только один поток.

Способы обеспечения взаимного исключения:

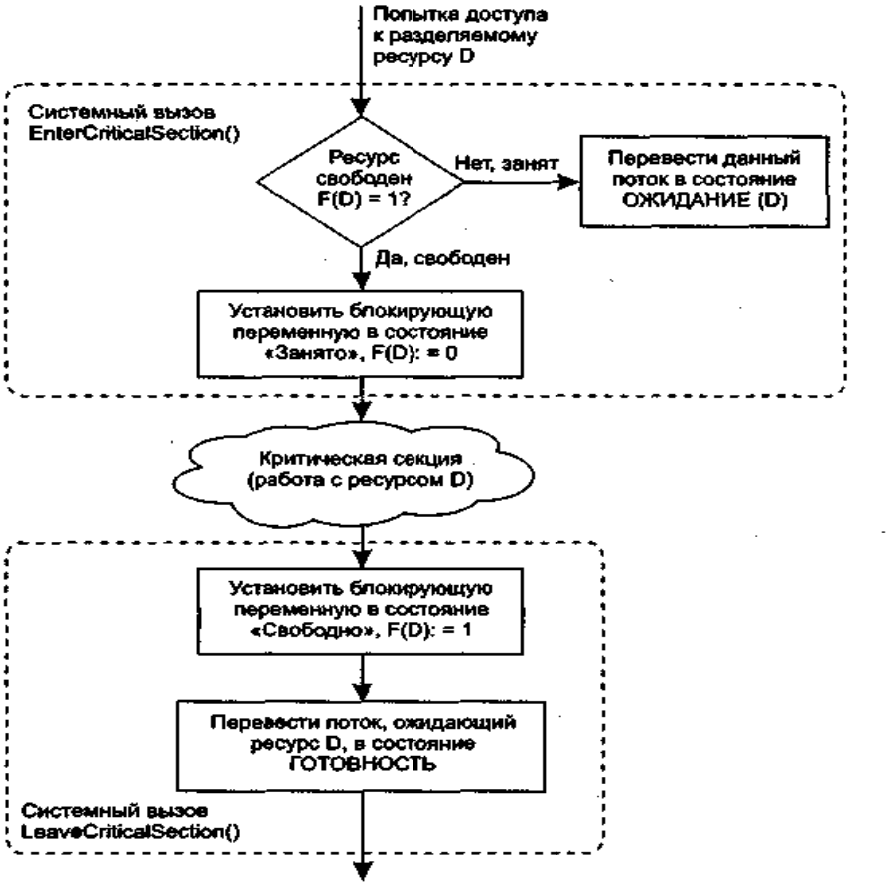
1. Простой, но неэффективный способ – запретить любые прерывания процесса во время его нахождения в критической секции (не применяется, т.к. означает потерю контроля над процессом).
2. Использование глобальных блокирующих переменных – программист, не обращаясь к ресурсам ОС, объявляет глобальные переменные, отражающие состояния критических данных. Обычно используются двоичные переменные, процесс устанавливает значение переменной равной 0, когда он входит в критическую секцию, и равной 1, когда он покидает критическую секцию.

В аппаратно-современных процессорах реализована неделимая операция – «проверка – установка».



Недостаток этого способа: бесполезная трата выделяемой процессу кванта времени в случае если ресурс занят. Процесс потратит на циклическое выполнение опроса переменной.

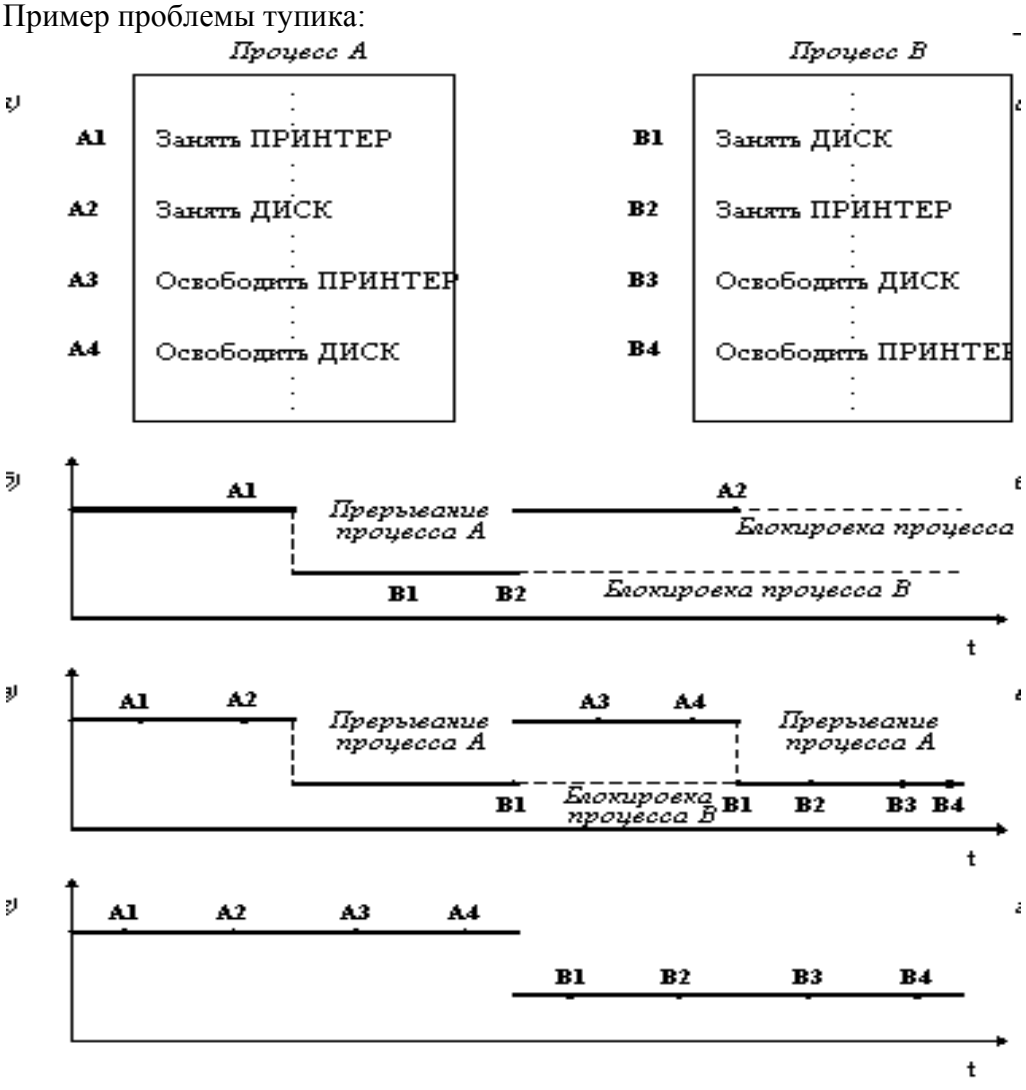
3. ОС предоставляет специальные системные вызовы для работы с критическими секциями, в рамках вызова выполняется проверка блокирующей переменной и если ресурс занят – поток переводится в состояние ожидания. ОС делает отметку о том, какой ресурс он ожидает и как только другой поток освобождает ресурс – вернет ожидающий процесс в состояние «готовность».



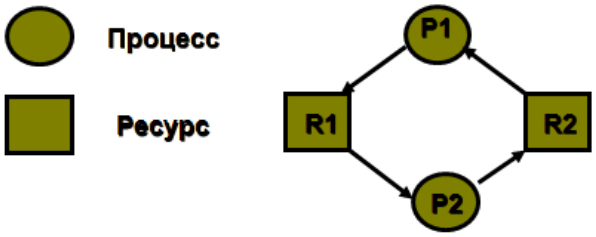
4. Обобщением блокирующих переменных являются семафоры. Они могут быть использованы для ресурсов, выделяемых дискретными частями. Для работы с семафорами вводятся два примитива, традиционно обозначаемых V и P. Пусть переменная S – семафор.
V(S): переменная S увеличивается на единицу. Выбор на наращивание и запоминание – не могут быть прерваны.
P(S): уменьшение S на единицу неделимых действий. Если S=0, то поток, вызвавший операцию P, ждет пока уменьшение станет возможным. Если max S=1, то семафор превратится в блокирующую переменную.
Одной из классических проблем синхронизации – задача о «читателях и писателях». Разделяемым ресурсом является буферный пул. Для работы с ним создаются 2 семафора: отражающий количество свободных буферов; отражающий количество занятых буферов. Поток «писатель» может работать только если имеются свободные буферы, а поток «читатель» - если имеются занятые.
Семафор – является универсальным средством синхронизации и может быть использован для синхронизации доступа к ресурсам любых видов:
- набор внешних устройств идентичного назначения (принтеры, порты);
- набор буферов в памяти;
- набор информационных структур.

21. Понятие тупиковой ситуации при выполнении параллельных вычислительных процессов

Пример проблемы тупика:



Имеются 2 вида разделяемых ресурсов: диск и принтер. Во избежание эффекта гонок, с каждым из них связана блокирующая переменная. Обеспечен принцип взаимного исключения.
В зависимости от скорости процессов они могут либо работать независимо (в), либо образовывать очередь к ресурсу (б), либо взаимно блокировать друг друга (а).
Группа процессов находятся в тупиковой ситуации, если каждый процесс из группы ожидает событие, которое может вызвать только другой процесс из этой же группы.
Условия возникновения тупиковых ситуаций:
1. Условие взаимного исключения. Каждый ресурс в данный момент или отдан одному процессу, или доступен.
2. Условие удержания и ожидания. Процессы в данный момент, удерживающие полученные ранее ресурсы, могут запрашивать новые ресурсы.
3. Отсутствие принудительной выгрузки ресурсов. У процесса нельзя забрать принудительно ранее полученные ресурсы.
4. Условие циклического ожидания. Существует круговая последовательность из двух и более процессов, каждый из которых ждет доступа к ресурсу, удерживаемый следующим членом последовательности.



Стратегия борьбы с взаимоблокировками:
1. Пренебрежение проблемой в целом.
2. Обнаружение и устранение взаимоблокировок (восстановление).
3. Недопущение тупиковых ситуаций с помощью аккуратного распределения ресурсов.
4. Предотвращение с помощью структурного опровержения одного из 4х условий, необходимых для взаимоблокировки.

22. Методы борьбы с тупиками.

Методы обнаружения взаимоблокировок.
Для обнаружения тупиков используется теория графов, ОС ведет таблицы свободных и занятых ресурсов, а также отмечает все поступившие запросы на ресурсы.
Математически это может быть описано:

$P = \{P_1, P_2, \ldots, P_n\}$ – множество процессов, n – число процессов;

$E = \{E_1, E_2, \ldots, E_m\}$ – множество ресурсов, m – число типов ресурсов;

$A = \{A_1, A_2, \ldots, A_m\}$ – вектор свободных ресурсов; $A_J <= E_J$, $j = \overline{1, m}$;

$C = \{c_{IJ} \mid i = \overline{1, n};\ j = \overline{1, m}\}$ – матрица текущего распределения ресурсов;

$R = \{r_{ij} \mid i = \overline{1, n};\ j = \overline{1, m}\}$ – матрица запрашиваемых ресурсов.

Существующие ресурсы

$E = \{E_1, E_2, \ldots, E_m\}$

$$\begin{bmatrix} c_{11} & c_{12} & \ldots & c_{1m} \\ c_{21} & c_{22} & \ldots & c_{2m} \\ \cdot & \cdot & \ldots & \cdot \\ c_{n1} & c_{n2} & & c_{nm} \end{bmatrix}$$

Доступные ресурсы

$A = \{A_1, A_2, \ldots, A_m\}$

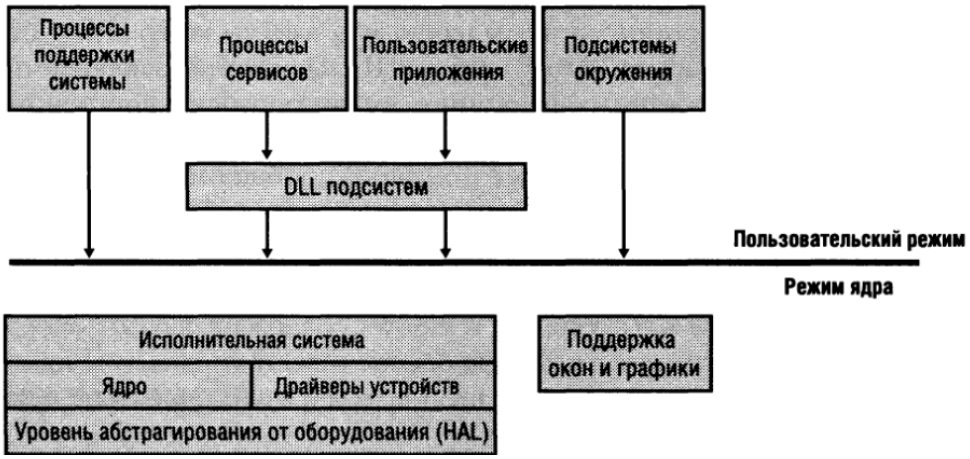
$$\begin{bmatrix} r_{11} & r_{12} & \ldots & r_{1m} \\ r_{21} & r_{22} & \ldots & r_{2m} \\ \cdot & \cdot & \ldots & \cdot \\ r_{n1} & r_{n2} & \ldots & R_{nm} \end{bmatrix}$$

$$\sum_{I=1}^{n} C_{ij} + A_j = E_j, \; j = \overline{1, m}$$

Алгоритм обнаружения тупиков основан на сравнении векторов ресурсов. В исходном состоянии все процессы не маркированы. По мере реализации алгоритма на процессы будет ставиться отметка, обозначающая, что они могут закончить свою работу, т.е. не находятся в тупике. После завершения алгоритма любой немаркированный процесс находится в тупиковой ситуации.
Алгоритм:
1. Ищется процесс P_i , для которого i – я строка матрицы R меньше вектора A, т. е. $R_i <= A$ или $r_{ij} <= A_j$, $j = 1, m$.
2. Если такой процесс найден, он маркируется, и далее прибавляется i - я строка матрицы C к вектору A, т.е. $A_j := A_j + c_{ij}$, $j = 1, m$. Возврат к шагу 1.
3. Если таких процессов не существует, работа алгоритма заканчивается. Если есть немаркированные процессы, то они попали в тупик.
Другим способом избежать тупиков является выделение процессорных ресурсов определенной, установленной ОС последовательности.
Этот метод получил название «мониторы Хоара».
Мониторы – системные программы, имеющие право выделять ресурсы процессам в строго установленном порядке.

Методы устранения тупиков.
1. Принудительная выгрузка ресурсов. Изъятие ресурса у процесса, передача его другому процессу, а затем возврат ресурса. Т.о., что исходный процесс этого «не замечает».
2. Восстановление через откат.
Процессы периодически создают контрольные точки, позволяющие запустить процесс с предыстории. При возникновении тупика процесс занимающий необходимый ресурс «откатывается» к контрольной точке, после которой он получил ресурс. Если возобновленный процесс вновь попытается получить данный ресурс, он переводится в режим ожидания освобождения этого ресурса.
3. Восстановление путем уничтожения процессов.
Недопущение тупиков теоретически возможно, а практически реализовать невозможно.

23. Архитектура ОС Windows.



Процессоры семейства Pentium имеют 4 уровня привилегий. ОС Windows использует 2 уровня:
- нулевой;
- третий.

Третий уровень используется для работы приложений и компонентов ОС, оформленных в виде серверов и называются пользовательским режимом.
В нулевом (режиме ядра) – работают исполнительная система и другие привилегированные компоненты ОС.
Компоненты режима ядра:
ядро ОС – реализует наиболее фундаментальные операции:
- создание объектов ядра;
- синхронизация процессоров в многопроцессорной системе;
- планирование потоков;
- обработку исключений;
- обработку аппаратных прерываний.
Исполнительную систему образуют менеджеры (диспетчеры), каждый из которых отвечает за определенный вид ресурсов:
1. Менеджеры процессов и потоков – создают и завершают процессы и потоки, используя сервисы ядра.
2. Менеджеры памяти – реализуют механизм виртуальной памяти.
3. Менеджеры КЭШа – управляют кэшированием диска.
4. Менеджеры безопасности – реализуют проверку прав доступа к объектам.
5. Менеджеры локального вызова процедуры – реализуют поддержку сетевых функций.
В режиме ядра работают также: драйверы, обеспечивающие поддержку нескольких файловых систем; менеджеры ввода/вывода, реализующие аппаратно-независимый ввод/вывод путем подключения цепочки драйверов.
Уровень абстрагирования от оборудования (HAL) – Hal.dll – это библиотека, которая реализует низкоуровневый интерфейс с аппаратурой, через нее работают компоненты Windows и драйвера других компонентов. Существует много версий HAL под различные аппаратные платформы, подходящий уровень выбирается в процессе установки Windows.
Часть компонентов ОС работают в пользовательском режиме. Они запущены друг от друга и от пользовательских приложений. Работа из ведется в соответствии с концепцией «клиент-сервер».
К процессам поддержки системы относятся: процессы, отвечающие за консольные окна; менеджеры сеансов (выполняют инициализацию и создание переменного окружения для пользователя; процесс winlogon, управляющий входом и выходом пользователя в систему и загрузкой оболочки Windows.
Важную роль играет подсистема win32, с помощью которой осуществляется взаимодействие пользователя и других приложений с компонентами, работающими в режиме ядра.
Системный интерфейс реализован в виде набора динамический библиотек.
Архитектура Windows базируется на использовании концепции объекта. Выделяют объекты ядра, пользовательские объекты и объекты графики (меню, окна, шрифты и др.).
Объекты ядра – структуры данных, доступ к которым имеет только ядро ОС. К ним относятся: процесс, поток, открытый файл, отображаемый файл, канал, событие, семафор и мьютикс (mutix).
Объект ядра принадлежит ядру, а не процессу, создавшего его. Объекты могут использоваться совместно многими процессами. Для этого каждый процесс создает свой собственный дескриптор этого объекта.
Ядро поддерживает подсчет использованности каждого объекта и уничтожает его только в том случае, если этот счетчик равен 0.
У объектов ядра есть атрибуты защиты, которые могут использоваться для ограничения доступа к ним. Это основное свойство, отличающее объекты ядра от пользовательских объектов и объектов графики.

24. Процессы и потоки ОС Windows. Алгоритмы их планирования.

Процесс – исполняемый экземпляр приложения и набор выделенных ему ресурсов. Ресурсы включают в себя:
- виртуальное адресное пространство;
- системные ресурсы (растровые изображения, файлы, области памяти);
- модули процесса (основной модуль (exe), динамические библиотеки, драйверы, управляющие элементы);
- уникальный номер – идентификатор процесса;
- один или несколько потоков выполнения.

Поток – внутренняя составляющая процесса, которой выделяется процессорное время.
Поток включает:
1. Уникальный идентификатор.
2. Два стека (для режима работы ядра и для пользовательского режима)
3. Участок памяти для работы подсистем и библиотек.
4. Информацию о текущем состоянии регистров процессора.

Состояние регистров, содержимом стеков и памяти называют контекстом потока.

Алгоритм планирования процессов и потоков Windows.
В Windows реализована вытесняющая многозадачность на уровне потоков. Используется механизм квантования и приоритетов.
Квант не измеряется в единицах времени, а выражается целым числом. Когда потоку выделяется квант, соответствующая переменная устанавливается в максимальное значение. Всякий раз, когда возникает прерываение таймера, из кванта вычитается 3.
Когда счетчик станет равным 0 – поток исчерпал свой квант. ОС решает вопрос о выборе следующего потока на выполнение.
Частота срабатывания таймера зависит от аппаратной платформы.
В Windows определено 32 уровня приоритета и 2 класса потоков: системные потоки (потоки реального времени) – от 31 до 16; потоки с переменным приоритетом (пользовательские) – от 1 до 15.
Приоритет 0 – зарезервирован для системных целей (поток обнуления страниц).
Уровень приоритета каждого потока складывается из трех составляющих:
- класс приоритета процесса;
- уровень приоритета потока внутри класса;
- динамический уровень приоритета.

Например: динамически повышается на 2 единицы приоритет потока, с которым непосредственно работает пользователь. Базовый уровень пользовательских приоритетов равен 8.
Приоритет реального времени (уровень 24) используется для системных процессов, которые простаивают большую часть времени до возникновения некоторых событий и не мешают остальным процессам.
Процесс класса реального времени обладают абсолютным приоритетом; пользовательские процессы планируются в соответствии с принципом относительных приоритетов.

25. Состояния процессов и потоков ОС Windows.

1. Готовность – у потока есть все, но не хватает только процессора (пул потоков).
2. Первоочередная готовность. Для каждого процессора системы выбирается один поток, который будет выполняться следующим (самый первый поток в очереди). Когда условия позволяют, происходит переключение на контекст этого потока.
3. Выполнение. Поток выполняется процессором и покинет это состояние либо если он завершится, либо если появится более приоритетный поток или закончится квант времени, либо, если он ожидает какое-либо событие.

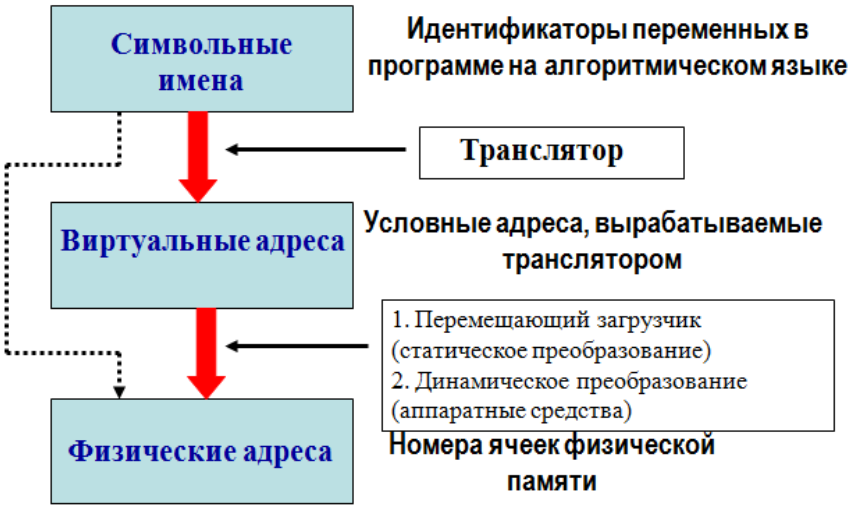
4. Ожидание. Поток может оказаться в этом состоянии либо по своей инициативе, если он ожидает некоторый объект для того, чтобы синхронизировать свое выполнение, либо ОС может ожидать чего-то в интересах потока, либо подсистема окружения может заставлять поток приостановить себя.
5. Переходное состояние. Поток входит в переходное состояние, если он готов к выполнению, но страница, содержащая стек потока, выгружена из ОП на диск в файл подкачки.
6. Завершение. Когда выполнение всех команд потока закончилось, он находится в состоянии завершения до тех пор, пока его не удалит менеджер объектов. Если в исполнительной части имеется указатель на этот же поток, то он может быть инициализирован и использован быть снова.

26. Функции ОС по управлению памятью. Типы адресов.

- Отслеживание и учет свободной и занятой памяти.
- Первоначальное и динамическое распределение памяти процесса приложения и самой ОС.
- Освобождение памяти при завершении процесса.
- Настройка адресов программы на конкретную область физической памяти.
- Полное или частичное вытеснение кодов и данных процессов из ОП на диск, когда размеры ОП недостаточны для размещения всех процессов и возвращение их в ОП.
- Защита памяти выделенной процессу от возможного вмешивания со стороны других процессов.
- Дефрагментация памяти.

Под памятью понимают оперативную память, а также внутренние регистры процессора и его КЭШ.
Память является неоднородной по своей структуре, отличаясь скоростью доступа и стоимостью в расчет на 1 бит.
Максимально доступный объем памяти с которой может работать ОС определяется ее разрядностью адреса.

Типы адресов.
Память адресуема, но для идентификации переменных и команд в различные периоды жизни процесса используются различные виды имен и адресов.



Достоинства и недостатки статического и динамического преобразований.
1. При статическом преобразовании исполняется программа перемещающий загрузчик, которая на основании имеющихся у нее исходных данных о начальном адресе физической памяти, в которой предстоит загрузить программу, и информации предоставлены транслятор, выполняющий загрузку программы совмещая и с заменой виртуальных адресов физическими.
Достоинством является высокая скорость и то, что преобразование выполняется один раз.
Недостаток – невозможность перемещения программы по ОП в процессе ее выполнения.
2. При динамическом преобразовании программа загружает в память в неизменном виде виртуальный адрес, но ОС фиксирует смещение действительного расположения программного кода относительно виртуального адресного пространства.
Во время выполнения программы при каждом обращении к памяти выполняется преобразование виртуального адреса в физический.
Например, путем сложения виртуального адреса и смещения.
Достоинство этого метода – программа жестко не прикрепляется к выделенному ей участку памяти.
Недостатки – значительное время на преобразование адресов.
В специализированных системах, когда заранее известно в какой области памяти будет выполняться система, транслятор выдает исполняемый код сразу в физических адресах.

27. Алгоритмы распределения памяти. Свопинг и виртуальная память.

Классификация методов распределения памяти.
Методы распределения памяти:
1. Без использования внешней памяти:
    1.1. Фиксированными разделами.
    1.2. Динамическими разделами.
    1.3. Перемещаемыми разделами.
2. С использованием внешней памяти:
    2.1. Страничное распределение.
    2.2. Сегментное распределение.
    2.3. Сегментно-страничное распределение.

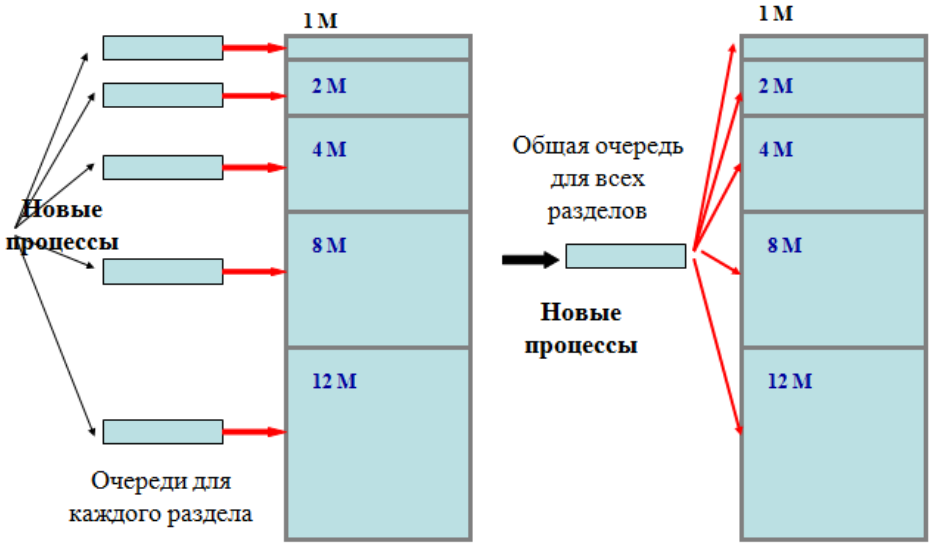
Методы распределения памяти с использованием внешней памяти называют виртуальной памятью. Виртуальным называется ресурс, который пользователю или его программе представляется обладающим свойствами, которыми он в действительности не обладает. Виртуальная память позволяет выполнить программы, размер которых превышает имеющуюся память.

Виртуализация ОП осуществляется с помощью программных модулей ОС и аппаратных схем компьютера.
Решаются следующие задачи:
1. Размещение данных в ЗУ разного типа (часть в ОП, часть на диске).
2. Выбор процессов и их частей для перемещения между ОП и диском.
3. Перемещение данных между памятью и диском.
4. Преобразование виртуальных адресов в физические. Разновидностью виртуальной памяти является swopping – перемещение между диском и ОП выполняется целиком для всего процесса.

Методы реализации виртуальной памяти:
1. Страничная виртуальная память – организует размещение данных между ОП и диском страницами – частями виртуального адресного пространства фиксированного и сравнительно небольшого размера.
2. Сегментная виртуальная память – предусматривает перемещение данных сегментами – частями виртуального адресного пространства произвольного размера, полученными с учетом значения данных.
3. Сегментно-страничная виртуальная память – использует двухуровневое деление: виртуальное адресное пространство делится на сегменты, а затем сегменты делятся на страницы. Единицей перемещения данных является страница.
4. Для временного хранения сегментов и страниц на диске отводятся специальная область – страничный файл или файл подкачки.
Механизм виртуальной памяти является прозрачным для программистов и пользователей.

28. Распределение памяти фиксированными, динамическими и перемещаемыми разделами.

Распределение памяти фиксированными разделами.
Память делится при старте системы на несколько разделов фиксированной величины. Процесс, поступивший на выполнение, помещается либо в общую очередь, либо в очередь к некоторому разделу.
Подсистема управления памятью выполняет следующие подзадачи:
1. Сравнивает размер программ, из свободных разделов выбирает подходящий раздел.
2. Выполняет загрузку программы и постройку адресов.

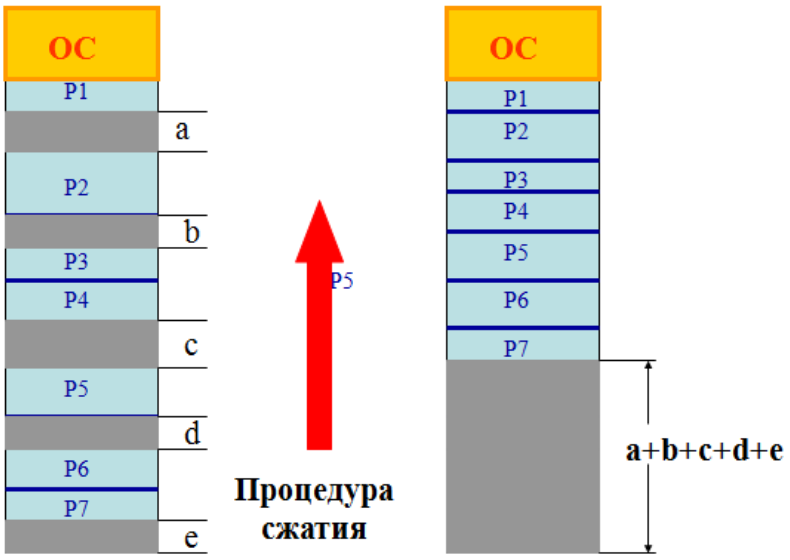


1. Разделы одинакового размера.
Недостатки:
- необходимость разработки оверлеев при больших размерах программ;

- неэффективное использование памяти (внутренняя фрагментация).
2. Разделы разного размера. Очередь к каждому разделу.
Достоинства: возможность распределения процессов между разделами с минимизацией внутренней фрагментации.
Недостатки: возможно неэффективное использование памяти засчет «простоя» больших разделов при наличии только небольших процессов.
3. Разделы разного размера. Общая очередь к разделу.
Достоинство: улучшается использование памяти, простота.
Достоинства распределения памяти фиксированными разделами: минимальные требования к ОС.
Недостатки: количество разделов, определенных во время генерации ОС, ограничивающих число активных процессов; неэффективное использование памяти.
Программист делит процесс на части называемые оверлеями. При запуске программы в память загружается нулевой оверлей, при завершении своей работы он вызывает первый оверлей и т.д.

Распределение памяти динамическими разделами.
Память не делится на разделы. Каждой поступившей задаче выделяется необходимый объем памяти, если достаточный объем памяти отсутствует, задача стоит в очереди. В произвольный момент времени помять представляет собой случайную последовательность занятых и свободных участков произвольного размера.
Достоинства: большая гибкость по сравнению с фиксированными разделами.
Недостаток: внешняя фрагментация.

Распределение памяти перемещаемыми разделами.
Функции ОС для реализации метода MVT OS/360:
- Ведение таблиц свободных и занятых областей ОП с указанием начального адреса и размера.
- При создании нового раздела просмотр таблиц и выбор раздела, достаточного для размещения процесса (наименьший или наиболее достаточный из свободных).
- Загрузка процесса в выделенный раздел и корректировка таблиц свободных и занятых областей основной памяти.
- После завершения процесса корректировки таблиц свободных и занятых областей.
Фрагментацией ОП называют наличие большого числа несмежных участков свободной памяти настолько маленького размера, что ни одна из стоявших в очереди программ не может быть размещена в памяти, хотя суммарный объем превышает требуемую величину.



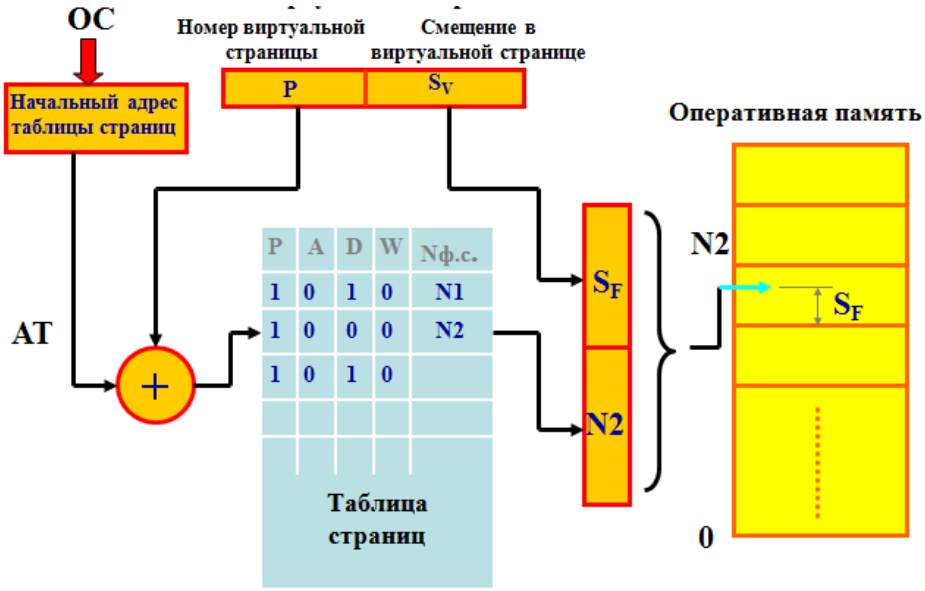
В дополнение к задачам ОС предыдущего метода необходимость периодически выполнять перемещение всех занятых участков в сторону младших или старших адресов. Кроме того:
- Изменение адресов команд и данных, к которым обращаются процессы при их перемещении в памяти за счет использования относительных адресов.
- Аппаратная поддержка процессором динамического преобразования относительных адресов в абсолютные адреса основной памяти.
- Защита памяти, выделяемой процессу, от взаимного внесения друг другу процессов.
Достоинства распределения памяти перемещаемыми разделами: эффективное использование ОП, исключение внутренней и внешней фрагментации.
Недостатки: дополнительные накладные расходы ОС.

29. Страничная организация виртуальной памяти.

Виртуальное адресное пространство процесса, полученное в результате трансляции программы делится на части одинакового фиксированного для данной системы размера, называемые виртуальными страницами. Объем страницы обычно выбирается равным 2^k/ Поскольку в общем случае размер адреса виртуальной памяти не кратен размеру страницы, поэтому последняя страница процесса дополняется фиктивной областью.
Вся ОП делится на часть такого же размера называется физическими страницами. Часть страниц помещается в виртуальную память, а остальные на диск.
Для каждого процесса ОС создает информационную структуру (таблицу страниц).
В таблице отображается соответствие между номерами виртуальных и физических страниц, или делается отметка о том, что виртуальная страница выгружена на диск.
В таблице страниц содержится также управляющая информация:
- признак модификации страницы;
- признак обращения;
- признак выгружаемости и т.д.
При каждом обращении к памяти происходит чтение из таблицы информации о соответствующей виртуальной странице.
Если страница находится в памяти, выполняется преобразование виртуального адреса в физический.
Если же нужная страница выгружена на диск, происходит так называемое страничное прерывание. Выполняющийся процесс переводится в состояние ожидания и активизируется другой процесс из очереди готовых, параллельно программа-обработчик прерываний находит на диске нужную страницу и пытается загрузить ее в ОП.
Если в памяти нет свободных страниц – решается вопрос (на основе принятой системе стратегии) какую страницу нужно выгрузить. У страницы, которая должна покинуть ОП, обнуляется ее бит присутствия и анализируется признак модификации.
Если страница была модифицированная на диск запишется ее новая версия.
Физически страница объявляется свободной, но в некоторых системах из соображений безопасности – свободная страница обнуляется.
Таблица страниц размещается в ОП, адрес таблицы страниц включается в контекст ее процесса.

Преобразование виртуальных адресов в физические.
Виртуальный адрес при страничном распределении может быть представлен в виде пары P, S_v: где P – виртуальный номер страницы, S_v – смещение в виртуальное странице.
Физический адрес может быть представлен в виде пары N, S_F: где N – номер физической страницы, S_F – смещение внутри физической страницы.
Учитывая, что объемы виртуальной и физической страниц равно, то $S_v=S_F$.

Поскольку объем страницы равен 2^k, смещение S может быть получено простым отделением k младших разрядов в двоичной записи адреса, а оставшиеся старшие разряды представляют собой двоичную запись номера страницы.
Начальный адрес страницы называют базовым адресом. Преобразование виртуального адреса в физический выполняется при помощи аппаратных схем процессора:
1. Из специального регистра процессора извлекается адрес АТ таблицы страниц. На основании начального адреса таблицы страниц, номера виртуальной страницы P (старшие разряды виртуального адреса) и длины отдельных записей таблицы (системная константа) определяется адрес нужной записи в таблице.
2. Извлекается номер соответствующей физической страницы N.
3. К номеру физической страницы присоединяется смещение S (младшие разряды виртуального адреса).
Использование операции присоединения вместо сложения позволяет повысить скорость преобразования.



Методы повышения эффективности функционирования страничной виртуальной памяти:
1. Структуризация ВОП, например, двухуровневая.
2. Хранение активной части записей таблицы страниц в высокоскоростном КЭШе или буфере быстрого преобразования адреса.
3. Выбор оптимального размера страниц.
4. Эффективное управление страничным обменом, использование оптимальных алгоритмов замены страниц.

Оптимальный размер страниц.
1. С уменьшением размера страницы уменьшается внутренняя фрагментация.
2. С уменьшением размера страницы увеличивается объем страничных таблиц и следовательно накладные расходы на работы ВП.
3. С увеличением размера страниц повышается скорость работы диска.
4. Частота страничных прерываний нелинейно зависит от размера страниц.

Управление страничным обменом.
Задачи управления:
- Когда передать страницу в основную память.
- Где разместить страницу в физической памяти.
- Какую страницу основной памяти выбирать для замещения, если в основной памяти нет свободных страниц.
- Сколько страниц процесса следует загрузить в основную память.

- Когда измененная страница должна быть записана во вторичную память.
- Сколько процессов разместить в основной памяти.

Идеальная стратегия замещения: вытеснять страницу ту, к которой дольше всего не будет обращений в будущем.
Наиболее распространенными алгоритмами являются:
1. Выбор дольше всего неиспользованной страницы.
2. Метод упреждающей загрузки.
3. Часовая стратегия замещения, основанная на выборе ближайшей неиспользующейся в настоящий момент страницы.

30. Сегментная организация виртуальной памяти.

Сегмент – это участок виртуального адресного пространства, размер которого определяется с учетом смыслового значения, содержащейся в нем информации.
При компиляции возможно создание следующих сегментов:
1. Исходный текст.
2. Символьная таблица, содержащая имена и атрибуты переменных.
3. Таблица констант.
4. Дерево грамматического разбора.
5. Стек, используемый для процедурных вызовов внутри компилятора.

По указанию программиста отдельными сегментами могут быть: массивы данных, процедуры и т.д.
На этапе создания процесса ОС создает таблицу сегментов процесса, в которой для каждого сегмента указываются:
1. Базовый адрес сегмента (начиная с которого загружается в ОП)
2. Размер сегмента.
3. Права доступа к сегменту.
4. Управляющая информация (признаки присутствия, модификации, исполнения, защиты и т.д.)

Если виртуальное адресное пространство нескольких процессов включает один и тот же сегмент, то в таблице сегментов этих процессов делаются ссылки на один и тот же участок ОП, в который этот сегмент загрузился в единственном экземпляре.
Механизм преобразования виртуального адреса в физический напоминает страничное преобразование, но требует гораздо больше времени, т.к. может быть получен только путем сложения базового адреса, взятого из таблицы сегментов и значения смещения внутри сегмента (младшие разряды виртуального адреса).
Достоинства сегментарной организации памяти:
1. Возможность совместного доступа пользователей к процедурам и данным.
2. Возможность задавать различные права доступа к различным сегментам.

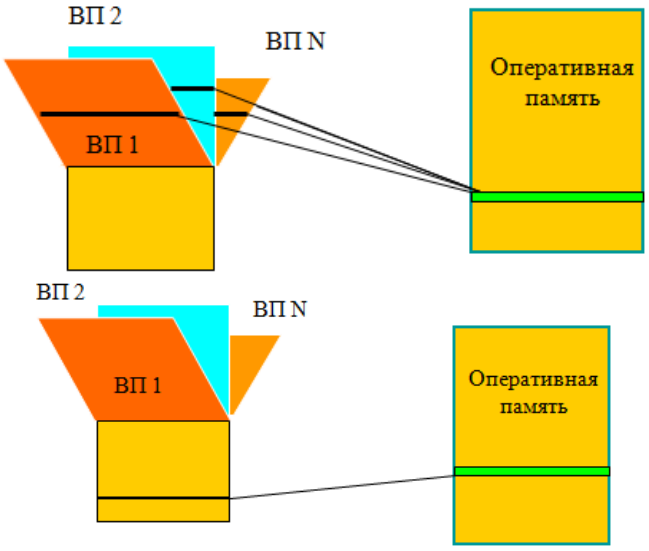
Недостатки:
1. Увеличение времени преобразования виртуального адреса в физический.
2. Избыточность перемещаемых данных (размер сегмента обычно больше размера страницы).
3. Внешняя фрагментация памяти (наличие участков памяти небольшой величины, куда невозможно поместить сегмент, хотя суммарно их величина значительная).

31. Сегментно-страничная организация виртуальной памяти. Разделяемые сегменты памяти.

Виртуальное адресное пространство делится на сегменты (позволяет задавать права доступа). Каждый сегмент делится на страницы. Физическая память делится на страницы. Перемещение между файлом подкачки и ОП выполняется постранично. Для каждого

процесса создается таблица сегментов, а для каждого сегмента – таблица страниц. Базовые адреса таблицы сегментов и таблицы страниц процесса являются частью его контекста и при активизации процесса загружаются специальные регистры процессора.
Преобразование происходит в 2 этапа:
1. Исходный виртуальный адрес преобразуется в линейный виртуальный адрес. Для этого вычисляется адрес дескриптора сегмента в таблице сегментов. Анализируются поля дескриптора и выполняется проверка возможности выполнения заданной операции. Если доступ разрешен, то вычисляется линейный виртуальный адрес путем сложения базового адреса в таблице страниц и номера страницы внутри сегмента.
2. Номер виртуальной страницы заменяется на номер физической и к нему присоединяется смещение внутри страницы.

Разделяемые сегменты памяти.
Хотя основной задачей ОС является защита областей памяти каждого процесса от доступа к ней остальных процессов, бывает необходимо организовать контролируемый совместный доступ нескольких процессов определенной области памяти.
Для организации разделяемого сегмента достаточно поместить его в виртуальное адресное пространство каждого процесса, которому он необходим, а затем настроить параметры отображения этих виртуальных сегментов так, чтобы они соответствовали одной и той же области физической памяти. Виртуальное адресное пространство каждого процесса может быть разделено на системную часть, в которой находятся используемые этим процессом ресурсы ОС, и индивидуальную часть.
Разделяемый сегмент может быть помещен либо в системную, либо в индивидуальную часть виртуального адресного пространства.



При работе с разделяемыми сегментами ОС должна выполнять некоторые функции общие для любых разделяемых ресурсов (файлы, семафоры и т.д.):
- поддержка именования ресурсов;
- проверка прав доступа;
- отслеживание количества процессов, использующих ресурс;
- удаление ресурса только в случае прекращения использования всеми процессами.

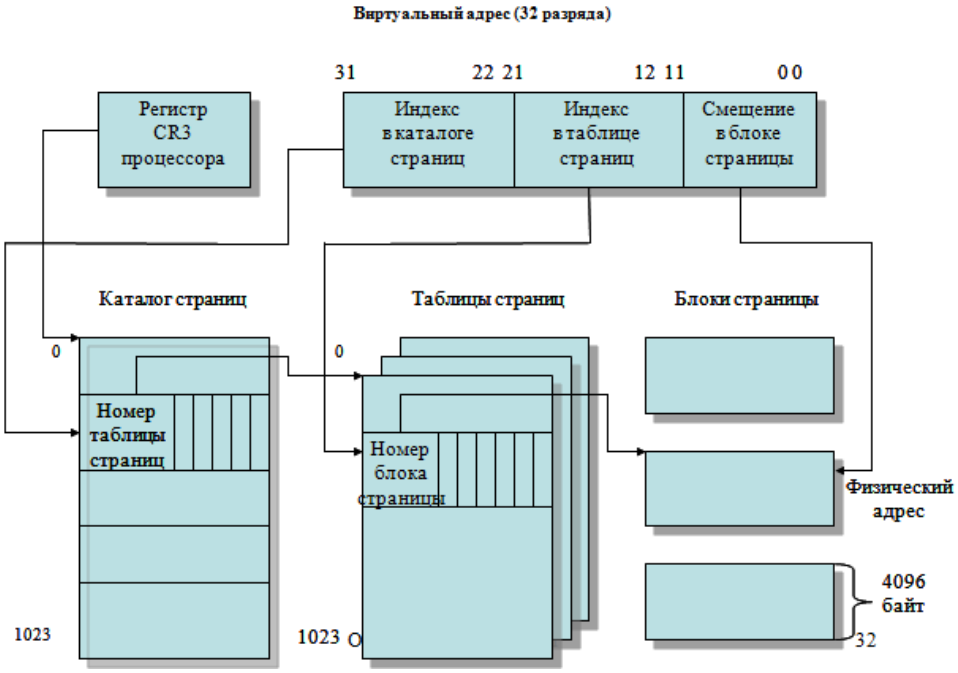
32. Архитектура памяти ОС Windows.

При использовании 32-разрядной версии Windows, виртуальная память обеспечит каждому процессу:
1. 4 Гбайт виртуального адресного пространства (2 Гб – ОС, 2 Гб – пользовательским программам).

2. 16 К независимых сегментов (8 К локальных и 8 К глобальных)
Младших 2 Гб доступно в пользовательском режиме. Сюда транслируются модули процесса; библиотеки; отображающиеся в память файлы; компоненты ОС выполняемые в пользовательском режиме. Пространство от 2 до 4 Гб зарезервировано для исполительной системы Windows, ядра и драйверов устройств, доступно только в привилегированном режиме.

33. Реализация страничного распределения в ОС Windows.

В 32-разрядной системе виртуальное адресное пространство процесса преобразуется в физический адрес следующим образом: для каждого процесса создается каталог страниц, номер записи в каталоге страниц содержится в старших разрядах виртуального адреса (с 31 по 22 адрес).
Для каждого раздела в каталоге страниц может быть создано 1024 таблицы страниц. Разряды виртуального адреса с 21-го по 12-й используются для хранения номера записи в таблице страниц.
Каждая таблица страниц содержит 1024 элемента, в которых хранятся номера физических страниц.
Добавляя к базовому адресу физической страницы младшие разряды виртуального адреса (с 11-го до 0-ой) содержащие смещение внутри страницы, можно получить физический адрес.
Преобразование виртуального адреса в физический: попадание.



Выделение памяти процессу происходит блоками страниц по 64 К.
Windows поддерживает также каталог системных страниц для работы с виртуальной памятью зарезервированной ОС.
Если страница находится в физической памяти, то в таблице страниц самый младший разряд (бит присутствия (достоверности)) устанавливается в единицу.
В случае, если страница отсутствует в оперативной памяти, этот бит устанавливается в 0 и разряды с 1-го по 4-ый интерпретируются как номер файла подкачки.
Windows поддерживает до 16 файлов подкачки.
Файл подкачки также поделен на страницы и номер страницы в этом файле хранится в старших адресах (с 31 по 12).

34. Размещение объектов в памяти. Куча и стек.

Регион (область или блок памяти) – непрерывная последовательность ячеек памяти.
Обычно регион больше области, а область больше блока.
В общем случае ОП, с которой работает программа, подразделяется на 3 вида:
- статическую;
- автоматическую;
- динамическую.

Статическая – область памяти, выделяемая при запуске программы до вызова функции main из свободной ОП для размещения глобальных и статических объектов, а также объектов, определенных в пространстве имен.
Автоматическая память – специальный регион памяти, резервирует при запуске программы до вызова функции main из свободной ОП и используется в дальнейшем для размещения локальных объектов (объектов, определяемых в теле функций и получаемых функциями через параметры в момент вызова). Автоматическая память часто называется стеком.
Динамическая память – совокупность блоков памяти, выделяемой из доступной свободной ОП непосредственно во время выполнения программы под размещение конкретных объектов.
Доступную программе свободную ОП называют кучей (heap). В общем случае нет четкого порядка расположения в ней, распределение блоков памяти происходит динамически, в большинстве реализаций под объект отводится первый подходящий по размеру свободный блок.
В отличии от стека, которым управляет компилятор, управление динамической памяти осуществляется явным образом, выделение памяти производится оператором new, освобождение – оператором delete.

35. Принцип действия кэш-памяти. Проблема согласования данных.

КЭШ-память – это способ совместного функционирования двух типов запоминающих устройств, отличающихся временем доступа и стоимостью хранения данных, которое за счет динамического копирования в «быстрое» ЗУ наиболее часто используемой информации из медленного ЗУ позволяется уменьшить среднее время доступа к данным.
Содержимое данной памяти представляет собой совокупность записей обо всех загруженных в нее элементов из основной памяти и каждая запись об элементе данных включает в себя:
1. Значение элемента данных.
2. Адрес, который этот элемент данных имеет в основной памяти.
3. Дополнительная информация, которая используется для реализации алгоритма замещения данных в КЭШе и обычно включает признак модификации и признак действительности данных.

КЭШ-память не является адресуемой, поэтому поиск нужных данных осуществляется по содержимому взятому из запроса значения поля адреса в ОП.
Далее возможен один из двух вариантов:
1. Если данные обнаружены в данной памяти, т.е. произошло КЭШ-попадание, они считываются из нее и результат передается источнику запроса.
2. Если нужные данные отсутствуют в КЭШ-памяти, то есть произошел КЭШ-промах, они считываются из основной памяти, передаются источнику запроса и одновременно с этим копируются в КЭШ.

Наличие в компьютере двух копий данных в основной памяти и КЭШе порождает проблему согласованности данных.

Рассмотрим два подхода к решению этой проблемы:
- сквозная запись;
- обратная запись.
Сквозная запись – при каждом запросе к основной памяти (в том числе при записи) просматривается КЭШ. Если данные по запрашиваемому адресу отсутствуют, запись выполняется только в основную память. Если же данные находятся в КЭШе, то запись выполняется одновременно в КЭШ и основную память.
Обратная запись: аналогично, если запрашиваемых данных нет в КЭШе, то запись выполняется только в основную память. В противном случае запись производится только в КЭШ-память, признак модификации устанавливается в единицу, что указывает на необходимость переписать их в основную память при вытеснении их из КЭШа.
В некоторых алгоритмах замещения предусмотрена первоочередная выгрузка модифицированных данных.

36. Способы отображения основной памяти на кэш.

Наиболее распространены две основных схемы отображения.
1. При случайном отображении, элемент ОП может быть размещен в произвольном месте КЭШ-памяти.
Для КЭШ со случайным отображением используется ассоциативный поиск, при котором сравнение выполняется параллельно со всеми записями КЭШа. Признак, по которому выполняется сравнение (адрес данных в ОП) называют тегом.
Аппаратная реализация такого способа достаточно дорогая, поэтому не используется для КЭШей большого объема.
2. Детерминированный способ. Предполагает, что любой элемент основной памяти всегда отображается в одно и тоже место КЭШа. В этом случае КЭШ разделен на строки, каждая из которых имеет номер. Между номерами строк КЭШ-памяти и адресами основной памяти устанавливается соответствие один ко многим.
В качестве отображающей функции может использоваться выделение несколькиз разрядов из адреса ОП, который интерпретируется как номер строки в КЭШ-памяти. Такое отображение называется прямым.
При поиске данных в КЭШе используется быстрый прямой доступ к записям по номеру строки, а затем выполняется проверка на совпадение старшей части адреса.
При совпадении тега фиксируется КЭШ-попадание.
Во многих современных процессорах используется комбинирование прямого и случайного отображения.
КЭШ делится на группы. Все группы пронумерованы, поиск в КЭШе выполняется сначала по номеру группы, полученного из адреса ОП, а затем в пределах группы путем ассоциативного просмотра на предмет совпадения страших частей адреса в ОП.
В современных процессорах используется несколько уровней КЭШа: для универсальных компьютеров до трех уровней. КЭШ уровна (n+1) больше по размеру и медленнее, чем КЭШ уровня n. КЭШ первого уровня является частью процессора, расположен на одном с ним кристалле и входит в состав функционирующих блоков. Состоит из КЭШа команд и КЭШа данных.
Латентность доступа КЭШа первого уровня равна 2-4 такта ядра.
КЭШ второго уровня может располагаться либо на кристалле, либо в наборе микросхем на системной плате. И его латентность от 8-20.
При работе с многоуровневым КЭШем запись данных из основной памяти сначала выполняется в КЭШ самого низкого уровня. При нахождении данных в КЭШе n-го уровня происходит их перезапись в КЭШ (n-1)-го уровня.

Кэширование является универсальным способом взаимодействия ЗУ с различной скоростью доступа (например, ОС использует часть ОП в качестве КЭШа для операций с динамической и флэш-памятью).

37. Управления вводом-выводом.

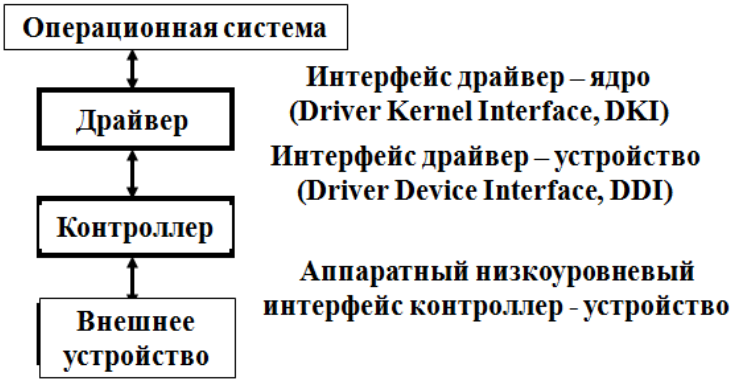
Для работы с устройствами ввода/вывода основным является следующий принцип: любые операции по управлению вводом/выводом объявляются привилегированными и могут выполняться только кодом ОС.
В современных ОС выделяют компонент – менеджер ввода/вывода. Его основные задачи:
1. Менеджер процессов получает запросы от прикладных задач на выполнение любых операций, в том числе на ввод/вывод. Эти запросы проверяются на корректность.
2. Менеджер ввода/вывода получает от некоторых процессов запрос на ввод/вывод.
3. Менеджер ввода/вывода вызывает соответствующее распределение каналов и контроллеров и планирует очередность устройств. Запрос выполняется немедленно, либо становится в очередь.
4. Этот менеджер передает управление соответствующим драйверам и в случае управления ввода/вывода с использованием прерываний передает информацию менеджеру процессов.
5. Этот менеджер осуществляет передачу сообщений об ошибках (если они происходят)
6. Менеджер ввода/вывода посылает сообщения о завершении операции ввода/вывода запросившей эту операцию задаче. Т.о. прикладные программы не могут непосредственно связываться с устройствами ввода/вывода, а только вызывают стандартные операции и устанавливают параметры, определяющие требуемые операции и количество потребляемых ресурсов.

При взаимодействии процессора с устройствами ввода/вывода используются устройство управления – контроллер (или адаптер).
Контроллер принимает команды и данные, а затем монопольно осуществляет управление устройством. После того, как команды выполнены либо произошла ошибка, контроллер генерирует сигнал-прерывание, о том, что процессор может передать следующую порцию данных и команд, или прочитать результаты.
Программа, называемая драйвером, учитывает особенности работы конкретного устройства и является посредником между контроллером и ОС.
Устройства ввода/вывода могут быть выделены в монопольное использование или быть разделяемыми. Чтобы организовать параллельную работу с монопольным устройством (например, принтер) вводится понятие виртуальных устройств.

Понятие виртуального устройства связано с концепцией спулинга.
Для записи потока данных отводится специальный файл на диске – спул-файл.
Системные процессы которые управляют спул-файлом могут выполнять операцию чтения и записи. В ОС Windows различают термины - принтер и устройство печати.
Принтер – объект ОС, а устройство печати – физическое устройство, подключающееся к компьютеру. Принтер может быть локальным, сетевым. При установке локального принтера, в ОС создается новый объект, связанный с устройством печати.
Локальность принтера означает, что его спул-файл будет находиться на том же компьютере, что и принтер. Если же компьютер предоставляет к сети общий доступ, то для другого компьютера он становится сетевым.
Компьютер, к которому подключен принтер становится принт-сервером.

38. Функции подсистемы ввода-вывода.

1. Организация параллельной работы ввода/вывода и процессора.
Существует 3 вида взаимодействий процессора с устройствами ввода/вывода:
- Программируемый ввод/вывод без прерываний (процессор посылает команды контроллеру и переводит процесс в состояние ожидания/завершения операции ввода/вывода).
- Ввод/вывод управляемый прерываниями (процессор посылает необходимые команды контроллеру ввода/вывода и продолжает выполнять процесс, если нет необходимости в ожидании выполнения операции. В противном случае процесс останавливается до получения прерывания, а процессор переключается на выполнение другого процесса).
- Прямой доступ к памяти (модуль прямого доступа к памяти управляет обменом данных между основной памятью и контроллером ввода/вывода. Процессор посылает запрос на передачу блока данных модулю прямого доступа к памяти, а прерывание происходит только после передачи всего блока данных).
2. Согласование скоростей обмена и кэширование данных.
Эта задача решается засчет буферизации данных в ОП и синхронизации доступа процессов к буферу. Кроме того, используются большие объемы буферной памяти в некоторых контроллерах внешних устройств.
3. Разделение устройств данных между процессами.
Устройства ввода/вывода могут предоставляться как в монопольное, так и в разделяемое использование, при этом ОС обеспечивает контроль доступа стандартными способами.
4. Обеспечение удобного логического интерфейса между устройствами и остальной частью системы.
В качестве такого интерфейса используется файловая модель устройства ввода/вывода. Для программиста устройство представляется последовательным набором байт, с которыми можно работать с помощью стандартных системных вызовов чтения и записи.
5. Поддержка широкого спектра драйверов с возможностью простого включения в систему нового драйвера.
ОС должна предоставлять возможность разработчикам устройств создавать драйвера.



Подсистема ввода/вывода может поддерживать несколько типов интерфейсов: DKI, DDI. Для поддержки различных драйверов параллельно с выпуском ОС выпускается набор инструментальных средств для отладки драйверов.
6. Динамическая загрузка и выгрузка драйверов.
7. Поддержка нескольких файловых систем.
Обычно в ОС имеется специальный слой ПО, экранирующий от пользовательских программ особенности физического хранения файлов.
8. Поддержка синхронных и асинхронных операций ввода/вывода.
Синхронный режим означает, что программный модуль приостанавливает свою работу до тех пор, пока операция ввода/вывода не будет завершена.
При асинхронном режиме – модуль продолжает выполняться в многозадачном режиме одновременно с операцией ввода/вывода.

39. Многослойная модель подсистемы ввода-вывода.

Многослойное построение ПО и его иерархичная структура позволяет с одной стороны учесть особенности каждого устройства, а с другой стороны – обеспечить единое логическое представление и унифицировать интерфейс всех типов устройств.
ПО делится не только на горизонтальные слои, но и на вертикальные.
Все устройства и их драйвера делятся на 2 класса:
1. Блок-ориентированные.
2. Байт-ориентированные (символьные).
Блок-ориентированные устройства – устройства прямого доступа, которые хранят информацию в блоках фиксированного размера, каждый из которых имеет собственный адрес (диск).
Байт-ориентированные устройства – не адресуемые, они генерируют или потребляют последовательность байт (клавиатура, терминал, сетевые адаптеры).
Наряду с модулями, отражающими специфику внешних устройств, в ОС существуют модули универсального назначения, образующие оболочку, называемые менеджерами ввода/вывода.
Верхний слой менеджера ввода/вывода образует системные вызовы, которые принимают запросы от пользовательских процессов и передают, отвечающим за определенный класс устройств, модулям и драйверам.
Нижний слой менеджера – взаимодействует с контроллером внешних устройств посредством прерываний.
Кроме того, этот менеджер взаимодействует с другими модулями ОС (менеджер процессов, менеджер памяти).
Выделяют низкоуровневые драйвера (аппаратные), которые непосредственно взаимодействуют с контроллером внешних устройств и вызывается по прерываниям, и высокоуровневые драйверы, которые оформляются по тем же правилам, но не вызываются по прерываниям, а предоставляют некоторый интерфейс и служат посредниками для передачи данных
Менеджер ввода/вывода создает цепочку драйверов для выполнения конкретной операции.

40. Цели и задачи файловой системы.

Файловая система – это часть ОС включающая:
1. Совокупность всех файлов на различных носителях информации (магнитные диски, магнитные листы и т.д.).

2. Наборы структур данных, используемых для управления файлами (каталоги и дескрипторы файлов, таблиц, распределение свободного и занятого пространства носителя информации).
3. Комплекс системных программных средств, реализующих различные операции над файлами (создание, чтение, запись, уничтожение, изменение свойств и т.п.)

41. Типы файлов. Иерархическая структура файловой системы. Имена файлов. Атрибуты файлов.

Все файловые системы поддерживают несколько типов файлов, основными из которых являются:
1. Обычные файлы (содержат информацию произвольного характера, заносимую пользователем или образующуюся в результате работы программ. Содержимое файла не контролируется ОС и определяется приложением, которое с ним работает. Двоичные файлы могут быть исполнительными, в этом случае ОС не требует наличия приложения.).
2. Каталоги (особый тип файла, который содержит системную справочную информацию о наборе файлов. В каталог могут входить файлы любых типов, в том числе и другие каталоги. Каталог устанавливает соответствие между именами файлов и их характеристиками (тип файла, расположение файла на диске, права доступа, дата создания и т.д.).
3. Специальные файлы – фиктивные файлы, ассоциированные с устройствами вывода. Позволяют пользователю выполнять операции ввода/вывода с помощью обычных команд записи в файл и чтение из файла. Эти команды сначала обрабатываются программой файловой системы, а затем преобразуются в команды управления соотвествующим устройством.
Современные файловые системы поддерживают также другие типы файлов: именованные контейнеры, символьные связи, проецируемые в память файлы.
Большинство файловых систем имеют иерархическую структуру, в которой уровни создаются за счет того, что каталог более низкого уровня может входить в каталог более высокого уровня.
Граф описывающий иерархию каталогов может быть представлен деревом или сетью.
Каталог верхнего уровня называют корневым. Частным случаем иерархической структуры является одноуровневая организация, когда все файлы входят в один каталог.
Все типы файлов имеют имена. Используются 3 типа имен:
1. Простое (символьное) – идентифицирует файл в пределах каталога. ОС накладывает ограничения на количество и номенклатуру использующуюся в имени символов.
2. Полное имя (составное) – представляет собой цепочку символов имен всех каталогов через которые проходит путь от корневого каталога до данного файла. В древовидной файловой системе между файлом и его полным именем устанавливается соответствие один файл – одно полное имя, а в сетевых – один файл – множество полных имен.
3. Относительное – определяется через понятие текущий каталог. Файловая система фиксирует имя текущего каталога и использует его как дополнение относительным именам для обращения полного имени файла.
Кроме символьного имени, ОС присваивает файлу уникальное имя, представляющее собой числовой идентификатор.
Понятие «файл» включает в себя не только имя и данные, но и атрибуты – информацию, описывающие свойства файла. Набор атрибутов определяется спецификой файловой системы, например, в однопользовательской ОС нет атрибутов – создатель файла, пароль доступа и т.д.

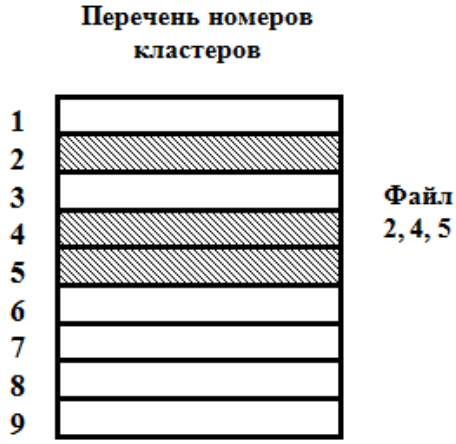
42. Физическая организация файловой системы.

Дисковый накопитель состоит из набора пластин, покрытых магнитным материалом. На каждой стороне пластины размечены концентрические кольца-дорожки (нумерация с 0 от внешнего края к центру).
Совокупность дорожек одного радиуса на всех пластинах называется цилиндром. Каждая дорожка радиально разбита на фрагменты, называемые секторами или блоками.
Каждый сектор может содержать равное количество байт (обычно 512), уплотнен к центру.
Контроллер находит сектор по адресу: номер цилиндра, номер поверхности, номер сектора.
Разметка дорожек и секторов выполняется на этапе физического и низкоуровнего форматирования.
На этапе высокоуровнего логического форматирования – диск может быть разбит на несколько разделов, каждый из которых может быть отформатирован под свою файловую систему.
ОС использует для размещения данных единицу дискового пространства, называемую кластером. Размер кластера от 1 до 2^k секторов.
На диске может располагаться несколько разделов, один из которых будет отмечен как активный – с него выполняется загрузка ОС.
Информация обо всех разделах, их размерах и статусе хранится в главной загрузочной записи (MBR), находящейся в нулевом секторе нулевой дорожки.
Кроме того, там же содержится программа NSB – внесистемный загрузчик.

43. Способы физической организации файла.

Можно выделить следующие способы физической организации файлов:
1. Непрерывное размещение.
Достоинства: высокая скорость доступа, минимальный объем адресной информации (номер первого кластера и объем файла), не ограничений на размер файла.
Недостатки: нет возможности для изменения размера файла, высокая степень возможной внешней фрагментации.
Область применения – компакт-диски.
2. Размещение в виде связанного списка кластеров. В начале каждого кластера содержится указатель на следующий кластер.
Достоинства: минимальная адресная информация, отсутствие внешней фрагментации, возможность изменения размера файла.
Недостатки: медленный доступ, сложность доступа к произвольному блоку файла (чтобы прочитать n-й кластер надо прочитать (n-1)-й, предыдущий, кластер), некратность блока файла степени двойки.
3. Связный список индексов
Часть дискового пространства отводится под хранение адресной информации. Каждому кластеру соответствует элемент в таблице, содержащей информацию о состоянии кластера.

Достоинства: быстрый доступ к произвольным кластерам файла, полное заполнение кластеров, кратное степени двойки.
Недостатки: рост адресной информации с увеличением емкости диска.
4. Перечень номеров кластеров: в информационной структуре хранится перечисленные номера кластеров, выделенные файлу.
Достоинства: высокая скорость доступа к произвольному кластеру благодаря прямой адресации, отсутствие внешней фрагментации.
Недостатки: длина адреса зависит от размера файла и может быть значительной.



44. Файловая система FAT.

Существует несколько версий файловой системы FAT: FAT12, FAT16, FAT32.
Основное их различие – разрядность используемых адресов кластеров.
Раздел диска, отформатированный под FAT имеет следующую структуру:
1. Загрузочный сектор.
2. Основная копия FAT.
3. Резервная копия FAT.
4. Корневой каталог – занимает фиксированную область и позволяет хранить 512 записей.
5. Область данных.
Поддерживается 2 типа файлов: обычный файл и каталог. Каталог представляет собой структурированный файл, содержащий следующие поля:
- имя файла (8 байт);
- расширение файла (3 байта);
- атрибуты файла (1 байт);
- зарезервировано (1 байт);

- время создания (3 байта);
- дата создания;
- дата последнего доступа;
- зарезервировано;
- время последней модификации;
- размер файла;
- начальный кластер файла.

Индексный указатель в таблице FAT может принимать следующие значения, характеризующие состояние связанного с ним кластера:
1. Кластер свободен (0).
2. Кластер выделен файлу, но не является последним кластером файла. В этом случае индексный указатель хранит номер следующего кластера файла.
3. Последний кластер файла (EOF).
4. Дефектный кластер (bad).
5. Резервный кластер.

При размещении файла ОС ищет 1-й свободный элемент в таблице FAT, записывает его в поле «начальный кластер». Если файлу нужно более одного кластера, создается цепочка (возможно, фрагментированная).
При удалении файла вместо 1-го символа в имени файла записывается признак «запись каталога свободна». Во все элементы таблицы FAT, отведенные файлу записывается признак «кластер свободен». В области данных никаких изменений не выполняется, что дает возможность восстановить файл.

45. Файловая система NTFS.

Основные свойства файловой системы NTFS.
1. Поддержка больших файлов и больших дисков (объем до 2^{64} байт)
2. Восстанавливаемость после сбоев и отказов программ и аппаратуры управления дисками.
3. Высокая скорость операций, в том числе для больших дисков.
4. Низкий уровень фрагментации, в том числе для больших дисков.
5. Гибкая структура допускающая развитие за счет добавления новых типов записей и атрибутов файлов с сохранением совместимости с предыдущими версиями файловых систем.
6. Устойчивость к отказам дисковых накопителей.
7. Поддержка длинных символьных имен.
8. Контроль доступа к каталогам и отдельным файлам.

Файл NTFS – не просто линейная последовательность байтов, характерная для FAT систем и Unix, а множество атрибутов, представляемых в виде потока байтов. Файл имеет несколько коротких потоков (имя, идентификатор и др.) и один или несколько длинных потоков с данными.
Структура тома NTFS.
Основой структуры тома является главная таблица файлов (Master File Table, MFT), которая содержит одну или несколько записей для каждого файла тома и одну запись для самой себя (размер записи 1, 2 или 4 кб).
Том состоит из последовательности кластеров, порядковый номер кластера в томе – логический номер кластера.
Файл состоит из последовательности кластеров, порядковый номер кластера внутри файла называется виртуальным номером кластера. Размер кластера от 512 байт до 64 кб.
Базовая единица распределения дискового пространства – отрезок (непрерывная область кластеров).
Адрес отрезка – (LCN, n), n – количество кластеров в отрезке.

Адрес файла (или его части) – (LCN, RCN, n).
Файл целиком размещается в записи таблицы MFT (если позволяет размер). В противном случае, в записи MFT хранится резидентная часть файла (некоторые его атрибуты), а остальная часть файла хранится в отдельном отрезке тома или нескольких отрезках.
Загрузочный блок содержит стандартный блок параметров BIOS, количество блоков в томе, начальный логический номер кластера основной и зеркальной копии MFT.
Структура файлов NTFS.
Файлы и каталоги состоят из набора атрибутов. Каждая запись MFT состоит из заголовка, за которым следуют атрибуты. Атрибуты содержат следующие поля: тип, длина, имя и значение.
Атрибуты используемые в записях MFT:
1. Стандартная информация (сведения о владельце, флаговые биты, время создания, время обновления и т.п.).
2. Имя файла в кодировке Unicode.
3. Список атрибутов (содержит ссылки на номера записей MFT, где расположены атрибуты), используется для больших файлов.
4. Версия – номер последней версии файла.
5. Дескриптор безопасности – список прав доступа ACL.
6. Версия тома – используется в системных файлах тома.
7. Имя тома.
8. Битовая карта MFT – карта использования блоков тома.
9. Корневой индекс – используется для поиска файлов в каталоге.
10. Размещение индекса – нерезидентная часть индексного списка (для больших файлов).
11. Идентификатор объекта – 64-разрядный идентификатор файла, уникальный для данного тома.
12. Данные файла.
13. Точка повторного анализа (монтирования и символьная ссылки)
Файлы NTFS в зависимости от способа размещения делятся на небольшие, большие и сверхбольшие.

46. Контроль доступа к файлам.

При определении прав доступа действует следующая схема. Пользователи, группы пользователей или процессы, запускаемые от их имени, являются субъектами доступа и пытаются выполнить некоторые операции над ресурсами-объектами доступа.
Объектами доступа могут быть любые разделяемые ресурсы: файлы, каталоги, области памяти и т.д. Для каждого типа объектов существует набор операций, которые с ним можно выполнить. У каждого объекта доступа существует владелец. Им может быть как отдельный пользователь, так и группа. Владелец объекта имеет право выполнить с ним любые допустимые для данного объекта операции.
Существует также особый пользователь (администратор в Windows, суперюзер в Unix), который имеет все права к любым ресурсам системы, не являясь при этом их владельцем.
В общем случае права доступа могут быть описаны матрицей прав доступа. Столбцы соответствуют всем файлам системы, строки – пользователям, а на пересечении указываются разрешенные операции.
В современных ОС матрица прав доступа хранится по частям. Например, в файловой системе NTFS для каждого файла хранится список прав доступа, в котором перечислены пользователи и группы, имеющие права работать с этим файлом.
Различают 2 подхода к назначению прав доступа:
1. Избирательный доступ – для каждого объекта сам владелец может определить допустимые операции с объектом. Между пользователем и группой нет четкой иерархии. Администратор наделен всеми правами.

2. Мандатный доступ – такой подход при котором система наделяет пользователя правами, в зависимости от того, к какой группе он относится. Владельцы объектов не имеют возможности управлять доступом к объектам. Все группы образуют строгую иерархию. Каждая группа пользуется правами группы более низкого уровня + права своего уровня. Членам группы не разрешается предоставлять права свои членам групп более низкого уровня.

Мандатные системы доступа более надежны и применяются в специализированных системах с повышенными требованиями к защите информации. В системах общего назначения используются избирательный подход.

В ОС семейства Unix существует 3 допустимых операции по отношению к файлам и каталогам:

1. Чтение (r).
2. Запись (w).
3. Выполнение (x).

Существует 3 объекта доступа:

1. Владелец файла.
2. Пользователи, входящие в группу владельца.
3. Все остальные.

По отношении. к каждому субъекту доступа определяются права, что дает запись из 9 символов, отсутствие права записывается как «-»

Например, rwx,r-x,r--.

Проверка разрешения доступа производится по следующей схеме:

1. При входе пользователя в систему для него создается token доступа, включающий в себя идентификатор пользователя и идентификатор всех групп, в которые входит пользователь, список прав пользователя, права доступа по умолчанию.

Все объекты (файлы и т.д.) при создании снабжаются дескриптором безопасности.

2. Получив запрос от процесса запущенного пользователем на выполнение некоторого вида операций над объектом, ОС находит характеристики безопасности этого объекта и последовательно сравнивает все идентификаторы процесса с идентификаторами владельца файла и идентификатора пользователя в группе в элементах списка доступа к файлу.
3. В случае совпадения идентификаторов пользователя с элементами списка доступа процесс получает разрешение на выполнение требуемых операций.

Разрешение на операции Windows бывают индивидуальными и стандартными.

По отношению к файлу есть в индивидуальных разрешения: read(r), write(w), execute(x), delete(d), take ownership(o).

Индивидуальные разрешения объединяются в стандартные разрешения. Для файла:

Стандартные разрешения	Индивидуальные разрешения
No access	ни одного
read	r, x
change	r, w, x, d
full control	все

Для каталога определено 7 стандартных разрешений:

- No access;
- List;
- Read;
- Add;
-Add & Read;
- Change;
- Full control.

47. Физическая организация ФС UNIX.

Рассмотрим организацию файловой системы на примере s5. Раздел диска делится на следующие части:
1. Загрузочный блок.
2. Супер-блок и его характеристики.
Содержит: размер файловой системы, размер области индексных дескрипторов, список свободных блоков, список свободных индексных дескрипторов и др.
3. Область индексных дескрипторов: порядок расположения соответствует их номерам, индексный дескриптор однозначно характеризует файл и хранит всю информацию о нем: тип файла, список блоков на диске, в которых расположен файл; права доступа к файлу; временные характеристики: число ссылок на индексный дескриптор равное количеству символьных имен файла, размер файла в байтах.
4. Область данных, в которой расположены как объектные файлы, так и каталоги, в том числе и корневой, специальные файлы устройств.
При создании файла ему выделяется номер из списка свободных индексных дескрипторов, а при удалении файла – номер его индексного дескриптора возвращается в этот список.
Особенностью файловой системы является отделение имени файла от его характеристик. Запись о файле в каталоге состоит из двух полей:
- символьное имя файла;
- номер индексного дескриптора.
Недостаток данной файловой системы является снижение производительности на жестких дисках большого размера.
Для устранения этого недостатка была предложена файловая система ufs. В ней раздел диска состоит из повторяющихся несколько раз последовательности областей:
1. Загрузчик.
2. Супер-блок.
3. Блок группы цилиндр.
4. Область индексных дескрипторов.
5. Область данных.
В повторяющихся областях содержится супер-блока. Блок группы цилиндра описывается количество индексных дескрипторов и блоков данных, расположенных в данной группе.
Просмотр дескрипторов и данных файла не приводит к большим перемещениям магнитных головок диска.