1. Принципы фон Неймана

Архитектура Фон Неймана:

ЦП = АЛУ(арифметико-логическое устройство)+ регистры и устройство управления ОЗУ ВВОД/ВЫВОД

1) Принцип двоичного кодирования (Преимущество перед десятичной системой счисления заключается в том, что устройства можно делать достаточно простыми, арифметические и логические операции в двоичной системе счисления также выполняются достаточно просто.)
2) Принцип адресуемости памяти (Структурно основная память состоит из пронумерованных ячеек (номер ячейки - адрес); процессору в произвольный момент времени доступна любая ячейка. Это позволяет обращаться к произвольной ячейке (адресу) без просмотра предыдущих. Время доступа к ячейке не зависит от её адреса)
3) Принцип однородности памяти (в ОП компьютера хранятся и данные и команды)
   Гарвардская архитектура — архитектура ЭВМ, отличительными признаками которой являются:
   1. Хранилище инструкций и хранилище данных представляют собой разные физические устройства.
   2. Канал инструкций и канал данных также физически разделены.
4) Принцип программного управления
   Процессор находится в бесконечном цикле из двух половинок:
   •Выборка и декодирование – выбирает очередную команду из оперативной памяти и пытается её понять (декодирует)
   •Исполнение
   В процессоре есть специальный регистр IP– адрес следующей команды. (Без IP невозможны for, if и т.д., а возможны только линейные программы.)
   Команды перехода меняют IP.

Кэш - промежуточный буфер с быстрым доступом, содержащий информацию, которая может быть запрошена с наибольшей вероятностью.

Прерывания (Если произошло событие, нарушающее работу процессора, происходит прерывание, при этом текущее состояние процессора сохраняется, а в IP записывается другой адрес)

1. внутренние (события в самом процессоре как результат нарушения каких-то условий при исполнении машинного кода: деление на ноль или переполнение стека, обращение к недопустимым адресам памяти или недопустимый код операции)
   a. маскируемые — прерывания, которые можно запрещать установкой соответствующих битов в регистре маскирования прерываний;
   b. немаскируемые — обрабатываются всегда, независимо от запретов на другие прерывания. К примеру, такое прерывание может быть вызвано сбоем в микросхеме памяти.
2. внешние (порожденные от внешних устройств – ввода/вывода: сигнал от таймера, сетевой карты или дискового накопителя, нажатие клавиш клавиатуры, движение мыши)
3. программные (инициируются исполнением специальной инструкции в коде программы, используются для обращения к функциям встроенного программного обеспечения, драйверов и ОС)

До окончания обработки прерывания обычно устанавливается запрет на обработку этого типа прерывания, чтобы процессор не входил в цикл обработки одного прерывания. Приоритезация означает, что все источники прерываний делятся на классы и каждому классу назначается свой уровень приоритета запроса на прерывание. Приоритеты могут обслуживаться как относительные и абсолютные.

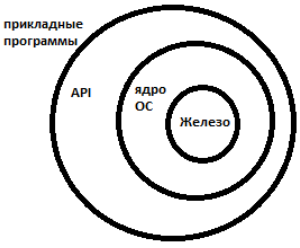
- Относительное обслуживание прерываний означает, что если во время обработки прерывания поступает более приоритетное прерывание, то это прерывание будет обработано только после завершения текущей процедуры обработки прерывания.
- Абсолютное обслуживание прерываний означает, что если во время обработки прерывания поступает более приоритетное прерывание, то текущая процедура обработки прерывания вытесняется, и процессор начинает выполнять обработку вновь поступившего более приоритетного прерывания. После завершения этой процедуры процессор возвращается к выполнению вытесненной процедуры обработки прерывания.

Вектор прерывания — закреплённый за устройством номер, который идентифицирует соответствующий обработчик прерываний. Векторы прерываний объединяются в таблицу векторов прерываний, содержащую адреса обработчиков прерываний. Местоположение таблицы зависит от типа и режима работы процессора.

2. Понятие и структура ОС

ОС – а) комплекс программ, обеспечивающий возможность управления оборудованием; б) расширенный компьютер(ВМ) (Архитектура большинства компьютеров на уровне машинных команд неудобна для использования прикладными программами, поэтому программисту удобнее работать с некими высокоуровневыми абстракциями – например, файл можно открывать для чтения или записи, использовать для получения или сброса информации, а потом закрывать. Это концептуально проще, чем заботиться о деталях перемещения головок дисков или организации работы мотора); в) менеджер ресурсов (во времени – в один момент времени только один принтер и в пространстве – распределение памяти на диске и т.п)
(Ресурсы – оборудование, время, файлы, безопасность, права пользователя (всё, от чего зависит работа компьютера, что можно дать или отнять))

Структура ОС:



Прикладные программы - программы, предназначенные для выполнения определенных пользовательских задач и рассчитанные на непосредственное взаимодействие с пользователем.

API (Application Program interface) – интерфейс прикладного программирования. (Набор системных функций и подпрограмм, посредством которых приложение получает доступ к операционной системе и другим сервисам. Прикладная программная среда)

Ядро ОС – главная часть ОС, работающая в привилегированном режиме (содержит планировщик; драйверы устройств, непосредственно управляющие оборудованием; сетевую подсистему, файловую систему;)

Реальная машина(железо) – компоненты компьютера (БИС и т.п)

Многозадачность и распределение полномочий требуют определённой иерархии привилегий компонентов самой ОС.

Есть 2 режима работы процессора:

- Привилегированный (режим ядра). В привилегированном режиме можно все

(выполнять любые команды и получать доступ к любой области памяти)
- Пользовательский. В пользовательском режиме ряд команд нельзя выполнять, и ограничен доступ к памяти.

3. Назначение, состав и функции ОС

Главное назначение ОС - это управление ресурсами, а главные ресурсы, которыми она управляет - это аппаратура компьютера, она управляет процессорами, памятью, устройствами ввода-вывода и данными. ОС также обеспечивает совместное функционирование всех устройств компьютера.

Функции

ОС реализует множество различных функций, в том числе:
- Прием от пользователя заданий, команд, сформулированных на соответствующем языке и их обработка
- Обеспечение загрузки пользовательских программ в оперативную память и их исполнение
- Распределение памяти, а в большинстве современных систем и организация виртуальной памяти
- обслуживание всех операций ввода-вывода
- организация параллельного выполнения двух и более программ на одном процессоре, создающая видимость их одновременного исполнения (многозадачность)
- защита одной программы от влияния другой, обеспечение сохранности данных, защита самой ОС от исполняющихся на компьютере приложений
- аутентификация и авторизация пользователей (многопользовательский режим)

Состав:
- BIOS –базовая система ввода вывода (выполняет тестирование основных компонентов при вкл., вызов начального блока загрузки ОС и передача ему дальнейшего управления, обработка прерываний)
- Загрузчик ОС (программа, которая находится в первом секторе жесткого диска, она выбирает, из какого из разделов жесткого диска следует продолжать загрузку, считывает основные дисковые файлы и передает дальней шее управление ЭВМ)

  Дисковые файлы IO.SYS(дополнения к BIOS компьютера) и MSDOS.SYS(реализует высокоуровневые услуги MS DOS)
- Ядро ОС (файловая система (отвечает за размещение информации на устройствах хранения); система управления памятью (размещает программы в памяти); система управления программами (осуществляет запуск и выполнение программ); система связи с драйверами устройств (отвечает за взаимодействие с внешними устройствами); система обработки ошибок; служба времени (предоставляет всем программам информацию о системном времени).)
- Командный процессор (принимает, обрабатывает команды, вводимые пользователем, и выполняет их).
- Внешние команды (поставляемые вместе с ОС в виде отдельных файлов)
- Драйверы устройств (программы, управляющие работой внешних (периферийных) устройств на физическом уровне. Они дополняют систему ввода-вывода ОС и обеспечивают обслуживание новых устройств или нестандартное использование имеющихся. Они передают или принимают данные от аппаратуры и делают пользовательские программы независимыми от ее особенностей.)

4. Классификация и примеры современных ОС

- ДОС (Дисковые Операционные Системы) (они представляют собой некий набор подпрограмм, не более того. ДОС загружает пользовательскую программу в память и передает ей управление, после чего программа делает с системой все, что ей хочется) *MS DOS*
- *Системы пакетной обработки* предназначались для решения задач в основном вычислительного характера, не требующих быстрого получения результатов. Главной целью и критерием эффективности систем пакетной обработки является максимальная пропускная способность, то есть решение максимального числа задач в единицу времени. Для достижения этой цели в системах пакетной обработки используются следующая схема функционирования: в начале работы формируется пакет заданий, каждое задание содержит требование к системным ресурсам; из этого пакета заданий формируется мультипрограммная смесь, то есть множество одновременно выполняемых задач. Для одновременного выполнения выбираются задачи, предъявляющие отличающиеся требования к ресурсам, так, чтобы обеспечивалась сбалансированная загрузка всех устройств вычислительной машины; так, например, в мультипрограммной смеси желательно одновременное присутствие вычислительных задач и задач с интенсивным вводом-выводом. Таким образом, выбор нового задания из пакета заданий зависит от внутренней ситуации, складывающейся в системе, то есть выбирается "выгодное" задание. Следовательно, в таких ОС невозможно гарантировать выполнение того или иного задания в течение определенного периода времени. В системах пакетной обработки переключение процессора с выполнения одной задачи на выполнение другой происходит только в случае, если активная задача сама отказывается от процессора, например, из-за необходимости выполнить операцию ввода-вывода. Поэтому одна задача может надолго занять процессор, что делает невозможным выполнение интерактивных задач. Таким образом, взаимодействие пользователя с вычислительной машиной, на которой установлена система пакетной обработки, сводится к тому, что он приносит задание, отдает его диспетчеру-оператору, а в конце дня после выполнения всего пакета заданий получает результат. Очевидно, что такой порядок снижает эффективность работы пользователя. *FMS (Fortran Monitor Sys.)*
- ОС общего назначения (рассчитанные на интерактивную работу одного или нескольких пользователей в режиме разделения времени) *Windows 2000, системы семейства Unix*
- Системы виртуальных машин (это ОС, допускающая одновременную работу нескольких программ, но создающая при этом для каждой программы иллюзию того, что машина находится в полном ее распоряжении, как при работе под управлением ДОС.) *MS DOS и MS Windows-эмуляторы для UNIX и OS/2*
- Системы реального времени. Применяются для управления различными техническими объектами. Способность гарантировать время реакции является отличительным признаком систем РВ. *программа бортового компьютера, управления спутником и т.п.*
- Системы промежуточных типов (системы, которые нельзя отнести к одному из вышеперечисленных классов.) Windows 95

Классификация:

1) Однозадачные. Операционная система поддерживает не более одного приложения запущенного в конкретный момент. *MS-DOS*

2) Многозадачные - ОС создаёт иллюзию нескольких одновременно рабочих приложений. В системе есть *системный таймер*, который управляет распределением процессорного времени между приложениями.
   2.1) Кооперативная многозадачность. Приложения сами отдают управление supervisor'у (центральный управляющий модуль ОС) тогда, когда сочтут нужным. Т.е. программа может забрать управление себе и не отдавать, после чего система превращается в однозадачную. Пример: Win 3.11
   2.2) Вытесняющая многозадачность. Управление отбирается у приложений по внешним прерываниям.
   2.2.1) Реального времени - гарантирует отклик на все события за фиксированное время.
   2.2.2) Разделения времени - управление переходит к supervisor'у по системному таймеру, после чего тот по своей "шаманской" формуле определяет, кому снова отдать управление. Пример: WinXP, Win9x.

Операционные системы могут различаться особенностями реализации внутренних алгоритмов управления основными ресурсами компьютера (процессорами, памятью, устройствами), особенностями использованных методов проектирования, типами аппаратных платформ, областями использования и многими другими свойствами, например:

Поддержка многопользовательского режима.

По числу одновременно работающих пользователей ОС делятся на:

однопользовательские (MS-DOS, Windows 3.x, ранние версии OS/2); *многопользовательские* (UNIX, Windows NT).

Главным отличием многопользовательских систем от однопользовательских является наличие средств защиты информации каждого пользователя от несанкционированного доступа других пользователей.

5 Понятие и назначение ядра ОС

Ядро-основная часть ОС, которая работает в привилегированном режиме.

Там находятся:

1. все, что связано в вводом, выводом
3. команды прерываний
4. команды системного управления процессором

Ядро представляет собой наиболее низкий уровень абстракции для доступа приложений к ресурсам системы, необходимым для их работы. Как правило, ядро предоставляет такой доступ исполняемым процессам соответствующих приложений за счёт использования механизмов межпроцессного взаимодействия и обращения приложений к системным вызовам ОС.

Все операции, связанные с процессами, выполняются под управлением этой части ОС.

Ядро обычно размещается в оперативной памяти, в то время как другие части ОС перемещаются во внешнюю память и обратно по мере необходимости.

Функции ядра

Ядро ОС, как правило, должно содержать программы для реализации следующих функций:

- обработка прерываний;
- операции над процессами (создание - завершение, блокировка - разблокировка, остановка – запуск, изменение приоритета);
- синхронизация процессов (приведение двух или нескольких процессов к такому их протеканию, когда определённые стадии разных процессов совершаются в определённом порядке, либо одновременно);
- организация взаимодействия между процессами;
- манипулирование блоками управления процессами;
- управление памятью
- поддержка операций ввода-вывода;
- поддержка работы файловой системы;
- поддержка механизма вызова-возврата при обращении к процедурам;
- ряд учетных функций.

6 Структура ядра современных ОС

Ядро состоит из трех основных подсистем:

- файловая подсистема;

Файловая подсистема контролирует права доступа к файлу, выполняет операции размещения и удаления файла, а также выполняет запись/чтение данных файла. Поскольку большинство прикладных функций выполняется через интерфейс файловой системы, права доступа к файлам определяют привилегии пользователя в системе.

Файловая подсистема обеспечивает перенаправление запросов, адресованных периферийным устройствам, соответствующим модулям подсистемы ввода/вывода.

- подсистема управления процессами и памятью;
  - создание и удаление процессов;
  - распределение системных ресурсов (памяти, вычислительных ресурсов) между процессами;
  - синхронизация процессов;
  - межпроцессорное взаимодействие.
  - отображение адресного пространства процесса на конкретные области физической памяти;
  - контроль доступа к адресным пространствам процессов;
  - учет свободной и занятой памяти
- подсистема ввода/вывода.

Подсистема ввода/вывода выполняет запросы файловой подсистемы и подсистемы управления процессами для доступа к периферийным устройствам (дискам, терминалам и т.п.). Она обеспечивает необходимую буферизацию данных и взаимодействует с драйверами устройств - специальными модулями ядра, непосредственно обслуживающими внешние устройства.

Также выделяют подсистемы:

- система управления программами (осуществляет запуск и выполнение программ);
- система связи с драйверами устройств (отвечает за взаимодействие с внешними устройствами);
- система обработки ошибок;
- служба времени (предоставляет всем программам информацию о системном времени).

7 Управление виртуальной памятью

Стратегии выделения ОП

1) best fit - помещается в тот раздел, где после его загрузки останется меньше всего свободного места
2) worst fit - При помещении в самый большой раздел в нем остается достаточно места для возможного размещения еще одного процесса
3) first fit - помещается в первый подходящий по размеру раздел.

Проблемы работы с физической памятью на прямую: защита процессов друг от друга и ОС от них, различное положение в памяти, при размещении нескольких процессов может не хватать непрерывной памяти.

Виртуальная память (Virtual Memory) представляет собой программно-аппаратное средство расширения пространства памяти, предоставляемой программе в качестве оперативной. Эта память физически реализуется в оперативной и дисковой памяти под управлением соответствующей операционной системы.

Виртуальная память была создана, потому что ОП не хватает, а так каждой программе, работающей в компьютере, выделяется сколько то виртуальной памяти. Реально это место в файле подкачки на жестком диске. (вместо слов: Реально ее не существует, её создает ОС).

[Эта память поделена на две части (чаще всего пополам): память программы (младшая) и память ядра и всех его таблиц (старшая). Из старшей памяти ничего нельзя даже считать.] – не знаю верно ли

Суть концепции виртуальной памяти заключается в том, что адреса, к которым обращается выполняющийся процесс, отделяются от адресов, реально существующих в физической памяти.

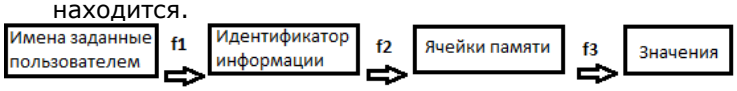
Несмотря на то, что процессы обращаются только к виртуальным адресам, в действительности они должны работать с реальной памятью, таким образом, во время выполнения процесса виртуальные адреса необходимо преобразовывать в реальные, причем это необходимо делать быстро, чтобы не снижалась производительность вычислительной машины.

Управлением виртуальной памятью называется процесс отображения виртуальной памяти системы в адресное пространство выполняющегося процесса.

Адресное пространство – размер всей возможной в архитектуре компьютера оперативной памяти.

Процесс можно представить, как отображение информации в память посредством трех функций.

- Именующей функции f1, однозначно отображающей данное пользователем имя в идентификатор информации, к которой это имя относится.
- Функции памяти f2, отображающей однозначно определенные идентификаторы в истинные адреса памяти, в которых они находятся.
- Функции содержимого f3, отображающей каждый адрес памяти в значение, которое по этому адресу находится.



Термин ***виртуальная память*** обычно ассоциируется с возможностью адресовать пространство памяти, гораздо большее, чем емкость первичной (реальной, физической) памяти конкретной вычислительной машины.

8. Стратегии подкачек и вытеснения страниц памяти

И виртуальная, и физическая (ОП) памяти разбиваются на *страницы одинакового размера* – это блок фиксированного размера.

Поэтому виртуальную память называют еще и страничной виртуальной памятью.

Виртуальный адрес представляется как пара: номер страницы и смещение (смещение необходимо для того, чтобы правильно определить виртуальный адрес).

Передача информации между памятью и диском всегда осуществляется целыми страницами. Так виртуальным адресам ставится в соответствие некий адрес физической памяти. Система отображения виртуальных адресов в физические сводится к системе отображения виртуальных страниц в физические и представляет собой *таблицу дескрипторов* (таблицу страниц).

Таблица страниц содержит:

1) Номер страницы
2) Реальный адрес
3) Флаг присутствия

Однажды может случится, что ОП закончится, тогда делается следующее:

ОС если поддерживает виртуальную память, то она имеет файл-подкачки (хранится на жестком диске). Среди страниц, которые заняты (в физической памяти) ядро произвольно выбирает страницу, копирует её в файл-подкачки, а страница становится свободной и тогда виртуальная память отображает страницу на освободившуюся. Если нам нужна страница, которая уже выброшена, то выбрасывается новая «жертва», а на освободившуюся страницу копируется страница из файла-подкачки (т.е. с жесткого диска).

Windows – pagefile.sys является файлом-подкачки.

Linux – не файл, а раздел диска swup. (жесткий диск работает медленнее ОП, поэтому это нежелательно)

Проблемы:

1) Узкое место фон Неймана – шина (замедление работы с памятью)

Решение: В современных ЦП – ассоциативная память, Хеш – поиск за один такт

2) Большой размер таблиц (в 32битной архитектуре – 32=20+12(номер страницы и смещение), 2^20=1млн записей у каждого процесса)

Решение: иерархическая таблица

3) Все страница физ.памяти заняты, а нам надо ещё.

Алгоритм выбора «жертвы»

1) идеальный алгоритм - выбор страницы, к которой дольше всего не будут обращаться, он не реализуем, но можно вести статистику и сравнивать (добавляем два поля R – признак чтения и M – модификации)
2) NRU. Not resently used - не использовавшаяся в последнее время страница, так же добавляем два поля в таблицу, там вроде 4 класса получается: 0 - не было обращений и изменений, 1 - не было обращений страница изменена, 2 - было обращение, страница не изменена, 3 - было обращение и изменение.
   Жертва- страница с наименьшим номером.
   Проблемы: Если хотим ввести счетчик обращения, то нужно помнить, что а) при каждом обращении счетчик менять нельзя (временные затраты, необходимость обновлять и отсортировывать таблицу), не покажет давнюю статистику (не хранит историю)
3) LRU - last recently used. Вместо счетчика как в NRU ввели таймер - то есть записывается время последнего обращения. Жертва - страница с наименьшим значением таймера. Наименьший - самый старый. Проблема – необходимость записи в таблицу. Модификация: хранить историю обращений.
4) NFU - не часто используемая страница(нечасто востребованная). Идея та же что и в LRU, реализуем с помощью счетчика. Проблемы - нет истории, и если страница много раз использовалась, а потом перестала использоваться, то ее нельзя удалить, так как значение таймера высокое. Можно улучшить: Модификация состоит из двух частей. Во-первых, перед добавлением к счетчикам значения бита R их значение сдвигается на один разряд вправо. Во-вторых, значение бита R добавляется к самому левому, а не к самому правому биту.
5) Рабочий набор страниц – те страницы, к которым программа обращалась последнее время. Если программа долго не работала и вышла из сна, то у нее не будет загруженных страниц, и следовательно, будет много прерываний. Поэтому многие системы замещения страниц пытаются отслеживать рабочий набор каждого процесса и обеспечивать его присутствие в памяти, перед тем как позволить процессу возобновить работу.
   Алгоритм: добавлены биты R и M, периодические прерывания от таймера запускают программу, очищающую бит обращения R. При каждой ошибке отсутствия страницы происходит сканирование таблицы страниц с целью найти страницу, пригодную для удаления.

При каждой обработке записи проверяется состояние бита R. Если его значение равно 1, текущее виртуальное время записывается в поле времени последнего использования таблицы страниц, показывая, что страница была использована при возникновении ошибки отсутствия страницы. Если обращение к странице происходит в течение текущего такта времени, становится понятным, что

она принадлежит рабочему набору и не является кандидатом на удаление. Если значение R равно 0, значит, за текущий такт времени обращений к странице не было и она может быть кандидатом на удаление. Измеряется ее возраст, сравнивается с контрольным значением. Если превосходит, то – стр-ца не относится к набору, заменяется новой. Если нет – то избегает удаления. В конце, если не найдут кандидата на удаление, то при наличии страниц с R=0, удалится одна из них с наибольшим возрастом. Иначе, удалится какая-то с R=1, предпочтение – неизмененным.

9. Программы, процессы и потоки

Программа — последовательность формализованных инструкций, предназначенная для исполнения устройством управления вычислительной машины, она записана на жестком диске.

Процесс –загруженная в память программа, подготовленная для исполнения.

С каждым процессом связывается его адресное пространство, из которого он может читать и в которое он может писать данные, пространство содержит: саму программу, данные к программе и стек программы.

Потоки – это работающий процесс, который выполняется.

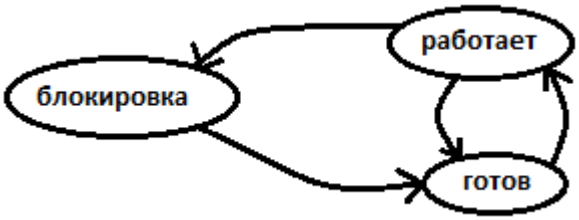
Поток = процесс + значение IP + регистр процессора + исполняемые файлы.

Выполняющимся (активным) потоком - называется поток управления, обрабатываемый в данный момент процессором. В многопроцессорных конфигурациях может одновременно выполняться несколько потоков.

Поток управления считается готовым к выполнению, если он способен стать активным, но не может этого сделать из-за отсутствия свободного процессора.

Процесс считается блокированным, если для продолжения его выполнения должно стать истинным некоторое условие, отличное от доступности процессора.

Состояния:



Завершение процесса – это завершение всех его потоков.

1. Нормальное завершение потока – все сделал
2. Нормальное завершение по ошибке (из-за невозможности дальше выполняться)
3. Фатальная ошибка, не предусмотренная программистом
4. Насильное завершение потока ОС

Один процесс может породить множество потоков, которые будут выполняться параллельно.

Параллельность достигается за счет разделения времени между потоками, это позволяет сделать системный таймер – это некий кварцевый генератор:

Происходит прерывание, попадает в ядро, в обработчик прерываний. В ядре счетчик, когда достигает определенного числа, сохраняет предыдущее состояние и переключается на другой поток. (еще бы про прерывания из 1)

Разделение времени зависит от приоритета, чем выше приоритет, тем больше времени дается. Кто долго работает должен иметь низкий приоритет.

Поток управления называют вытесненным, когда его выполнение приостановлено из-за того, что другой поток с более высоким приоритетом уже готов к выполнению.

10. Кооперативная и вытесняющая многозадачность

Многозадачность — свойство операционной системы или среды программирования, обеспечивать возможность параллельной (или псевдопараллельной) обработки нескольких процессов.

кооперативная многозадачность

Тип многозадачности, при котором следующая задача выполняется только после того, как текущая задача явно объявит себя готовой отдать процессорное время другим задачам. (Фоновые задачи выполняются только во время простоя основного процесса и только в том случае, если на это получено разрешение основного процесса.) При кооперативной многозадачности приложение может захватить фактически столько процессорного времени, сколько оно считает нужным. Преимущества кооперативной многозадачности: отсутствие необходимости защищать все разделяемые структуры данных объектами типа критических секций и mutex'ов, что упрощает программирование, особенно перенос кода из однозадачных сред в многозадачные. Недостатки: неспособность всех приложений работать в случае ошибки в одном из них, приводящей к отсутствию вызова операции «отдать процессорное время». Крайне затрудненная возможность реализации многозадачной архитектуры ввода-вывода в ядре ОС, позволяющей процессору исполнять одну задачу в то время, как другая задача инициировала операцию ввода-вывода и ждет её завершения.

Вытесняющая многозадачность (режим реального времени)

Вид многозадачности, в котором операционная система сама передает управление от одной выполняемой программы другой в случае завершения операций ввода-вывода, возникновения событий в аппаратуре компьютера, истечения таймеров и квантов времени, или же поступлений тех или иных сигналов от одной программы к другой. В этом виде многозадачности процессор может быть переключен с исполнения одной программы на исполнение другой без всякого пожелания первой программы и буквально между любыми двумя инструкциями в её коде. К тому же каждой задаче может быть назначен пользователем или самой операционной системой определенный приоритет, что обеспечивает гибкое управление распределением процессорного времени между задачами (например, можно снизить приоритет ресурсоёмкой программе, снизив тем самым скорость её работы, но повысив производительность фоновых процессов). Распределение процессорного времени осуществляется планировщиком процессов. Этот вид многозадачности обеспечивает более быстрый отклик на действия пользователя. Преимущества: возможность полной реализации многозадачного ввода-вывода в ядре ОС, когда ожидание завершения ввода-вывода одной программой позволяет процессору тем временем исполнять другую программу. Сильное повышение надежности системы в целом, в сочетании с использованием защиты памяти — идеал в виде «ни одна программа пользовательского режима не может нарушить работу ОС в целом» становится достижимым хотя бы теоретически, вне вытесняющей многозадачности он не достижим даже в теории. Возможность полного использования многопроцессорных и многоядерных систем. Недостатки: необходимость особой дисциплины при написании кода, особые требования к защите всех разделяемых и глобальных данных объектами типа критических секций и mutex'ов.

11. Планировщики задач с приоритетами

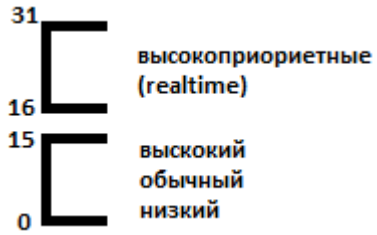
Когда компьютер работает в многозадачном режиме, на нем зачастую запускается сразу несколько процессов или потоков, претендующих на использование центрального процессора. Такая ситуация складывается в том случае, если в состоянии готовности одновременно находятся два или более процесса или потока. Если доступен только один центральный процессор, необходимо выбрать, какой из этих процессов будет выполняться следующим.

Та часть операционной системы, на которую возложен этот выбор, называется **планировщиком**, а алгоритм, который ею используется, называется **алгоритмом планирования**.

Т.к. сущ-ет три типа систем: пакетные (предназначались для решения задач в основном вычислительного характера, не требующих быстрого получения результатов. Главной целью и критерием эффективности систем пакетной обработки является максимальная пропускная способность, то есть решение максимального числа задач в единицу времени.), интерактивные (критерий эффективности - удобство и эффективность работы пользователя), реального времени (критерием эффективности здесь является способность выдерживать заранее заданные интервалы времени между запуском программы и получением результата), то сущ-ет несколько алгоритмов.

1) FIFO – первой обслуживается задача, которая первая поступила. Выполняемая задача->в состояние блокировки->в состояние готов->в конец очереди, как только поступившая. **+**: простота реализации и восприятия; **-**: временные затраты.
2) Rownd-Robin – круговая очередь. **Каждому процессу назначается определенный интервал времени, называемый его квантом, в течение которого ему предоставляется возможность выполнения.** Если процесс к завершению кванта времени все еще выполняется, то ресурс центрального процессора у него отбирается и передается другому процессу. Если завершился или перешел в сост-е блокировки до истечения кванта, то в этот момент передается дальше. Особенности: если время кванта – слишком большое, то переключение процесса – начасто, то есть только при логической необходимости (это +). Если маленькое – то снижается эффективность исп-я ЦП, слишком длинная – медленная реакция на короткие запросы (это -). Компромисс: 20-50мс.
3) Очередь с приоритетами - каждому процессу присваивается значение приоритетности, и запускается тот процесс, который находится в состоянии готовности и имеет наивысший приоритет.

Пример приоритетов в Windows
(32 градации)



Приоритеты

Приоритеты могут быть статическими или динамическими

Статические приоритеты не изменяются, такой механизм установки приоритетов достаточно прост и не сопряжен с большими издержками. Однако следует учитывать, что такой механизм недостаточно гибок, т.к. не реагирует на изменение окружающей ситуации.

Динамические приоритеты (в современных ОС) позволяют повысить реактивность системы, т.к. реагируют на изменения ситуации, и начальное значение приоритета процесса может быть изменено на новое, более подходящее значение. Он изменяется в зависимости от того, насколько активно задачу использует процессор и другие системные ресурсы. Предпочтительными для системы будут те программы, которые захватывают процессор на короткое время и быстро отдают его, переходя в состояние ожидания внешнего или внутреннего события. Таким процессам система стремится присвоить более высокий приоритет.

Таким образом, система динамически повышает приоритет тем заданиям, которые освободили процессор в результате запроса на ввод-вывод или ожидание события и, наоборот, снижает тем заданиям, которые были сняты по истечении кванта времени. (Чем дольше программа работает, тем меньше у нее приоритет, затем он резко возрастает до прежнего уровня и так циклично, пока работает поток)

12. Основные примитивы синхронизации потоков

Иногда необходимо передавать данные из одного процесса в другой (когда? – графический интерфейс и параллельное программирование (моделирование параллельными алгоритмами)).

Для программ, использующих несколько потоков или процессов, необходимо, чтобы все они выполняли возложенные на них функции в нужной последовательности. Для этой цели предлагается использовать несколько механизмов, обеспечивающих слаженную работу потоков (механизмы синхронизации).

Например, разрабатывается программа, в которой параллельно работают два потока. Каждый поток обращается к одной разделяемой глобальной переменной. Один поток при каждом обращении к этой переменной выполняет её инкремент, а второй – декремент. При одновременной асинхронной работе потоков неизбежно возникает такая ситуация: первый поток прочитал значение глобальной переменной в локальную; ОС прерывает его, так как закончился выделенный ему квант времени процессора, и передаёт управление второму потоку; второй поток также считал значение глобальной переменной в локальную, декрементировал её и записал новое значение обратно; ОС вновь передаёт управление первому потоку, тот, ничего не зная о действиях второго потока, инкрементирует свою локальную переменную и записывает её значение в глобальную.

Очевидно, что изменения, внесённые вторым потоком, будут утеряны.

Для исключения подобных ситуаций необходимо разделить во времени использование совместных данных. В таких случаях используются механизмы синхронизации, которые обеспечивают корректную работу нескольких потоков.

Средства синхронизации:

1) Критическая секция – фрагмент программы, зависящий от общих данных. Существует так же функции инициализации (создания) и удаления, вхождения и выхода из критической секции. Ограничения: поскольку это не объект ядра, то он не виден другим процессам, то есть можно защищать только потоки своего процесса (не может синхронизировать потоки разных процессов).
   Возможные решения: 1) запретить прерывания (внутри ОС); 2) ввести блокирующую переменную (что-то вроде флага, если неактивна, то поток входит в крит. секцию и устанавливает 1 в пер-ю, выполняет свои задачи и снова освобождает пер-ю (0)); 3) Ввести стадию активного ожидания (spinlock – что-то вроде флага: если занято – вертится в цикле, ожидая пока освободится, иначе – использует и потом освобождает.) Эффективность можно обеспечить – никакие потоки одновременно не находятся в крит.секции, потоки вне кр.секции не должны мешать другим войти в неё, никакой поток не должен бесконечно ждать входа в кр.секцию.
2) Mutex - это один из вариантов семафорных механизмов для организации взаимного исключения. Они реализованы во многих ОС, их основное назначение — организация взаимного

исключения для потоков из одного и того же или из разных процессов. Ни один другой поток не может завладеть мьютексом, который уже принадлежит одному из потоков. Если мьютекс защищает какие-то совместно используемые данные, он сможет выполнить свою функцию только в случае, если перед обращением к этим данным каждый из потоков будет проверять состояние этого мьютекса. Windows расценивает мьютекс как объект общего доступа, который можно перевести в сигнальное состояние или сбросить. Сигнальное состояние мьютекса говорит о том, что он занят. Потоки должны самостоятельно анализировать текущее состояние мьютексов. Если требуется, чтобы к мьютексу могли обратиться потоки других процессов, ему надо присвоить имя. С mutex возможны 2 операции: lock(m)- заблокировать и unlock(m)- разблокировать. (опять же флаг!) Это обуславливает поочередный доступ к общему ресурсу.

3) Семафор - объект, ограничивающий количество потоков, которые могут войти в заданный участок кода. Определение введено Эдсгером Дейкстрой (1965). Объект ядра "семафор" используется для учёта ресурсов и служит для ограничения одновременного доступа к ресурсу нескольких потоков. Используя семафор, можно организовать работу программы таким образом, что к ресурсу одновременно смогут получить доступ несколько потоков, однако количество этих потоков будет ограничено. Создавая семафор, указывается максимальное количество потоков, которые одновременно смогут работать с ресурсом (инициализировать счетчик =N). Каждый раз, когда программа обращается к семафору, значение счетчика ресурсов семафора уменьшается на единицу. Когда значение счетчика ресурсов становится равным нулю, семафор недоступен. (sem.enter()). Прекращение исп-я семафора – увеличение счетчика на 1.

Таким образом, синхронизация решается методом поочередной блокировки с помощью семафора и mutex.

Не все виды синхронизации решаются данным способом, поэтому существует второй вид – синхронизация по времени (должно произойти **некоторое событие**, которого дожидается переменная» q-war (Unix), event (Windows). С этим понятием можно использовать две операции: wait (E)- ждать сигнал и Signal (E)- послать сигнал, причем события не гарантируют, что потоки не проспят сигнал.

Единичное событие (single event) – это скорее общий флаг. Событие находится в сигнальном состоянии, если его установил какой-нибудь поток. Если для работы программы требуется, чтобы в случае возникновения события на него реагировал только один из потоков, в то время как все остальные потоки продолжали ждать, то используют единичное событие.

Мануальное событие (manual event) — это не просто общий флаг для нескольких потоков. Оно выполняет несколько более сложные функции. Любой поток может установить это событие или сбросить (очистить) его. Если событие установлено, оно останется в этом состоянии сколь угодно долгое время, вне зависимости от того, сколько потоков ожидают установки этого события. Когда все потоки, ожидающие этого события, получат сообщение о том, что событие произошло, оно автоматически сбросится.

Два примитива синхронизации:

1) Обмен сообщениями (Для решения проблемы согласованности процессов при передаче информации, существуют специальные механизмы обмена данными, основанные на мьютексах. Это системные вызовы send() и recieve(). Первый посылает сообщение заданному адресату, второй получает сообщение от указанного источника. Если сообщения нет, второй запрос блокируется до поступления сообщения либо немедленно возвращает код ошибки.)

2) Барьер — это определяемый пользователем примитив синхронизации, разрешающий нескольким потокам (называемым участниками) выполнять алгоритм параллельно и поэтапно. Каждый участник выполняется, пока его код не достигнет точки барьера. Барьер означает окончание одного этапа работы. Когда участник достигает барьера, его выполнение блокируется, пока все участники не достигнут этого же барьера. Когда все участники достигнут барьера, при необходимости можно вызвать действие следующего этапа. Это действие следующего этапа может использоваться для выполнения действий одним потоком, пока все остальные потоки все еще остаются блокированными. После выполнения этого действия блокировка всех участников снимается.

13. Проблема тупиков и способы борьбы с ней

Предположим, что несколько процессов конкурируют за обладание конечным числом *ресурсов*. Если запрашиваемый процессом *ресурс* недоступен, ОС переводит данный процесс в состояние ожидания. В случае, когда требуемый *ресурс* удерживается другим ожидающим процессом, первый процесс не сможет сменить свое состояние. Такая ситуация называется ***тупиком.***

Определение. Множество процессов находится в тупиковой ситуации, если каждый процесс из множества ожидает события, которое только другой процесс данного множества может вызвать. Одновременно в нем может находится как 1, так и несколько процессов.

Определение 2. Процесс или поток находится в состоянии тупика, если он ожидает какого-либо события, которое никогда не произойдет. Иногда подобные ситуации называют ***взаимоблокировками.***

Зависание системы (Dead Lock) – ситуация, когда один или несколько потоков находятся в состоянии тупика.

Условия возникновения тупиков сформулированы Коффманом, Элфиком и Шошани в 1970 г.

1. Условие взаимоисключения. Одновременно использовать ресурс может только один процесс.
2. Условие ожидания ресурсов. Процессы удерживают ресурсы, уже выделенные им, и могут запрашивать другие ресурсы.
3. Условие неперераспределяемости. Ресурс, выделенный ранее, не может быть принудительно забран у процесса. Освобожден он могут быть только процессом, который их удерживает.
4. Условие кругового ожидания. Существует кольцевая цепь процессов, в которой каждый процесс ждет доступа к ресурсу, удерживаемому другим процессом цепи.

В общем случае проблема *тупиков* эффективного решения не имеет.

Способы борьбы с взаимоблокировками

- Игнорирование проблемы в целом

(Подход большинства популярных ОС (Unix, Windows и др.) состоит в том, чтобы игнорировать данную проблему в предположении, что маловероятный случайный тупик предпочтительнее, чем нелепые правила, заставляющие пользователей ограничивать число процессов, открытых файлов и т. п.)

- Предотвращение *тупиков*

Можно избежать *взаимоблокировки*, например, используя ***алгоритм банкира***. Идея основана на том, что всегда есть резерв. Алгоритм базируется на так называемых **безопасных** или **надежных** состояниях. ***Безопасное состояние*** – это такое состояние, для которого имеется, по крайней мере, одна последовательность событий, которая не приведет к *взаимоблокировке*.

В отсутствие информации о будущих запросах единственный способ избежать взаимоблокировки – добиться невыполнения хотя бы одного из условий возникновения тупиков. Например, ***Нарушение условия ожидания дополнительных ресурсов***
Условия ожидания дополнительных ресурсов нарушается при двухфазном захвате по принципу «все или ничего». В первой фазе процесс запрашивает все необходимые ему ресурсы. До момента полного их предоставления процесс не может продолжить свою работу. Если некоторые ресурсы, которые нужны процессу и запрошены в первой фазе, заняты другими процессами, процесс освобождает все ресурсы, которые были ему выделены, и повторяет первую фазу. Недостатком метода является неэффективное использование компьютера. ***Нарушение принципа отсутствия перераспределения*** можно осуществить, если отобрать ресурсы у удерживающих их процессов до завершения работы этих процессов. Данный метод имеет множество недостатков. ***Нарушение условия кругового ожидания.*** Один из способов – упорядочивание ресурсов. Например, присвоение уникальных номеров всем ресурсами задание условия запроса процессами ресурсов в порядке их возрастания. Данное решение исключает круговое ожидание.

- Обнаружение *тупиков*

При обнаружении тупика следует зафиксировать тупиковую ситуацию и выявить вовлеченные в нее процессы. При выполнении первых трех условий возникновения тупика проверяется наличие циклического ожидания. Для обнаружения используют графы распределения ресурсов.

- Восстановление после *тупиков*

2 способа: 1) завершение выполнения одного или более процессов для использования их ресурсов, 2) временно забрать ресурс у текущего владельца и передать его другому процессу.
В ряде систем предусмотрены средства отката и перезапуска или рестарта с контрольной точки (сохранение состояния системы в какой-то момент времени).

14. Гармонически взаимодействующие процессы

Легко показать, что критические секции может иметь только программа, работающая с разделяемыми структурами данных. И этими критическими секциями как раз и являются места, где программа модифицирует такие структуры или просто обращается к ним. Синхронизация доступа к разделяемым структурам часто приводит к усложнению программы, а стремление сократить участки исключительного доступа - к ошибкам. Желание устранить эти проблемы привело в свое время Дейкстру к концепции, известной как гармонически взаимодействующие последовательные процессы.

По его мнению, «Гармоническое взаимодействие» - это когда процессы:
- либо не взаимодействуют
- либо взаимодействуют в строго определенных точках кода программ

Эта концепция состоит в следующем:

- Каждый процесс представляет собой независимый программный модуль, для которого создается иллюзия чисто последовательного исполнения.
- Процессы не имеют разделяемых данных.
- Все обмены данными, и вообще взаимодействие, происходят в выделенных точках процессов. В этих точках процесс, передающий данные, останавливается и ждет, пока его партнер будет готов эти данные принять. В некоторых реализациях процесс-передатчик может не ожидать приема, а просто складывать данные в системный буфер. Аналогично, процесс, принимающий данные, ожидает, пока ему передадут данные. Иными словами, все передачи данных неразрывно связаны с синхронизацией.
- Синхронизация, не сопровождающаяся передачей данных, просто лишена смысла - процессы, не имеющие разделяемых структур данных, совершенно независимы и не имеют ни критических точек, ни нерентабельных модулей.

Концепция гармонически взаимодействующих процессов очень привлекательна с теоретической точки зрения и позволяет легко писать правильные программы. Однако часто по соображениям производительности оказывается невозможно отказаться от разделяемой памяти.

15. Механизмы межзадачного взаимодействия

ОС предоставляет механизмы, облегчающие взаимодействие приложений и совместный доступ к данным. Возможности, обеспечиваемые этими механизмами, принято называть межзадачными (межпроцессными) взаимодействием (IPC -interprocess communications). Одни механизмы IPC позволяют нескольким специализированным процессам совместно выполнять обработку данных, а другие - распределенную обработку данных в сети.
Процессы не могут взаимодействовать не общаясь. Общение процессов обычно приводит к изменению их поведения в зависимости от полученной информации. Если деятельность процессов остается неизменной при любой принятой ими информации, то это означает, что они на самом деле не нуждаются во взаимном общении. Процессы, которые влияют на поведение друг друга путем обмена информацией, принято называть кооперативными или взаимодействующими процессами, в отличие от независимых процессов, не оказывающих друг на друга никакого воздействия и ничего не знающих о взаимном сосуществовании в вычислительной системе.
Рассмотрим основные три механизма:
1) Труба(Pipe - канал) – универсальный инструмент передачи информации между процессами. В общем случае канал можно представить в виде трубы, соединяющей два процесса. Что попадает в трубу на одном конце, мгновенно появляется на другом. Чаще всего каналы используются для передачи непрерывного потока данных.
(Это при условии, что в современных ОС программы изолированы друг от друга.)
Pipe – временный файл, позволяющий общаться программам.
Pipe бывают двух видов:

- Безымянные, анонимные (односторонние, только чтение и запись, строго последовательные)

Безымянные каналы используются для перенаправления стандартного ввода/вывода дочернего процесса так, чтобы он мог обмениваться данными с родительским процессом. Чтобы производить обмен данными в обоих направлениях, следует создать два безымянных канала. Родительский процесс записывает данные в первый канал, используя его дескриптор записи, в то время как дочерний процесс считывает данные из канала, используя дескриптор чтения. Аналогично, дочерний процесс записывает данные во второй канал и родительский процесс считывает из него данные. Безымянные каналы не могут быть использованы для передачи данных по сети и для обмена между несвязанными процессами.

- именованные (двусторонние, операции создать, открыть, закрыть, читать, решать и т. д., если труба пустая, то заблокируется пока туда ничего не запишут) В этом виде существует проблема, что имена совпадут, но универсального способа борьбы с этим не существует.

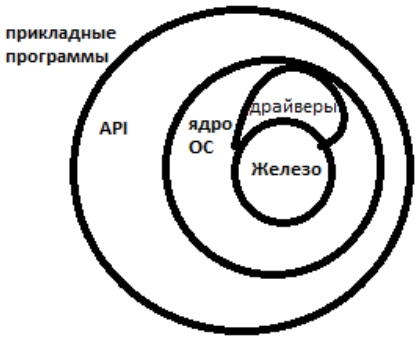
Именованные каналы являются объектами ядра ОС Windows, позволяющими организовать межпроцессный обмен не только в изолированной вычислительной системе, но и в локальной сети. Они обеспечивают

дуплексную связь и позволяют использовать как потоковую модель, так и модель, ориентированную на сообщения. Обмен данными может быть синхронным и асинхронным.
Именованные каналы используются для передачи данных между независимыми процессами или между процессами, работающими на разных компьютерах. Обычно, процесс сервера именованных каналов создает именованный канал с известным именем или с именем, которое будет передано клиентам. Процесс клиента именованных каналов, зная имя созданного канала, открывает его на своей стороне с учетом ограничений, указанных процессом сервера. После этого между сервером и клиентом создается соединение, по которому может производиться обмен данными в обоих направлениях. В организации межпроцессного обмена наибольший интерес представляют именованные каналы.
2) «почтовые ящики» (Unix – mailbox, Windows - mailslot)
«Почтовый ящик» – область ОП (32-64 Кб), поэтому быстрая, но маленькая. Почтовые ящики обеспечивают одностороннее взаимодействие процессов. Каждый процесс, который создает почтовый ящик, является «сервером почтовых ящиков» (mailslot server). Другие процессы, называемые «клиентами почтовых ящиков» (mailslot clients), посылают сообщения серверу, записывая их в почтовый ящик. Все входящие – в ящик, пока не прочитает сервер. Если ящик полон, то при новом письме выталкиваем самое старое. Mailslot является псевдофайлом, находящимся в памяти, и следует использовать стандартные функции для работы с файлами, чтобы получить доступ к нему. Данные в почтовом ящике могут быть в любой форме - их интерпретацией занимается прикладная программа, но их общий объем не должен превышать 64 Кб. Однако, в отличие от дисковых файлов, mailslot являются временными - когда все дескрипторы почтового ящика закрыты, он и все его данные удаляются. Все почтовые ящики являются локальными по отношению к создавшему их процессу; процесс не может создать удаленный mailslot.
3) Mapped file - файлы, проецируемые в память
Отображение файлов в память позволяет процессу интерпретировать содержимое файла так, как будто оно является блоком памяти в его адресном пространстве. Процесс может использовать простые операции для чтения и модификации содержимого файла. Когда два или более процесса получают доступ к одному и тому же отображенному в память файлу (данным), каждый процесс получает указатель на память в своем собственном адресном пространстве и может читать или модифицировать содержимое файла.
Отображение файла в память может быть использовано только для взаимодействия между процессами на одном компьютере, но не в сети. Таким образом, отображаемые в память файлы предоставляют возможность нескольким процессам получать совместный доступ к данным, однако процессы должны обеспечивать синхронизацию доступа к данным.

16. Классификация внешних устройств

1) Классификация по типу (значению)
   a) – ввода (клавиатура, мышь (+тачпад, трекболл, джостик, руль), сканер, капера, микрофон, графический планшет, xBox)
   b) – вывода (монитор, принтер, колонки, тактильные устройства)
   c) - устройства внешней памяти
      - Магнитные накопители (ленты, магнитные и жесткие диски, стримеры)
      - Оптические накопители (лазером, CD, DVD)
      - Флэш-устройства
   d) – связные устройства (сетевые карты, модемы, ИК-порт и т.п)
2) По передаче данных
   a) – символьные это устройства, которые умеют передавать данные только последовательно, байт за байтом (мышь, принтер, клавиатура, модем)
   b) – блочные - это устройства, которые могут передавать блок байтов как единое целое. (жесткий и гибкий диски)
3) По доступу
   a) – произвольные – имеют возможность обратиться к любому элементу последовательности за равные промежутки времени (дисковые накопители (виртуальный диск ОЗУ), флеш-устройства)
   b) – последовательное – доступ к группе элементов осущ-ся в заранее заданном порядке (стримеры, оптические диски – CD/DVD)

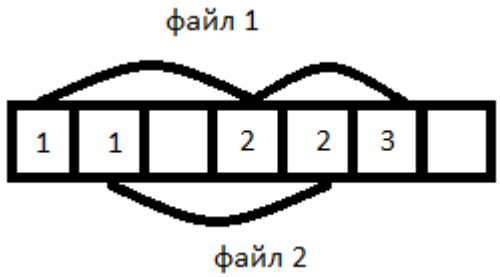
17. Драйверы внешних устройств.



Драйвер устройства – это программа, которая обеспечивает управление устройством, т.е. позволяющая конкретному устройству взаимодействовать с операционной системой.
Как драйверы подключаются к ОС?
1) Часть ядра («умолчания»)
Если устройство включено в список совместимого оборудования, то драйвер такого устройства обычно входит в состав Windows.
2) Динамически-загружаемый модуль.
Драйверы устройств загружаются автоматически при запуске компьютера и с этого момента выполняются, оставаясь невидимыми.
a) в Linux: insmod – загрузить модуль ядра, lsmod – показать модуль ядра
2) в Windows: Plug and Play – техника холодного подключения (автоматически ищет драйвера для подключённого устройства). Идея: воткнуть новое устройство, во время загрузки обнаруживается, ищется драйвер, устанавливается и все работает.
3) Горячая загрузка драйвера – USB.
Большинство ОС запрещают пользовательским программам непосредственный доступ к аппаратуре. Это делается для повышения надежности и обеспечения безопасности в многопользовательских системах.
Чаще всего драйверы являются частью ядра системы, исполняются в высшем кольце защиты и имеют доступ на запись к сегментам данных пользовательских программ, а часто и к данным самого ядра, т.е. драйверы всегда работают в режиме ядра. Отсюда следующие опасности:
- могут повредить работе ядра
- их установка позволяет нам влезть внутрь ядра
Это ведет к уменьшению безопасности.

18. Файлы, каталоги и файловые системы.

Файл – это именованная область диска.
Именованная – есть имя.
Адресное пр-во – все возможные имена.
1) Алфавит
2) Зависимость от регистра
3) Длина
4) Иерархия (плоская или с каталогами)
Область – упорядоченный набор секторов:
1) непрерывный (CD/DVD)



2) кусками:

Файл - это место постоянного хранения информации: программ, данных для их работы, текстов, закодированных изображений, звуков и др.

Структурная организация файлов:

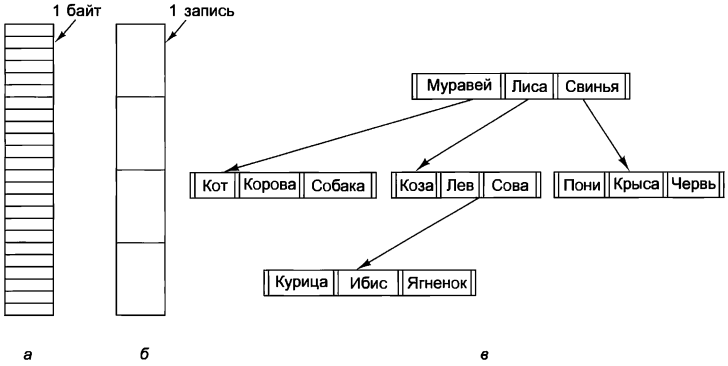


Рис. 4.1. Три типа файлов: последовательность байтов (*а*); последовательность записей (*б*); дерево (*в*)

Типы файлов:

Обычными считаются файлы, содержащие информацию пользователя. Каталоги — это системные файлы, предназначенные для поддержки структуры файловой системы. Символьные специальные файлы имеют отношение к вводу-выводу и используются для моделирования последовательных устройств ввода-вывода, к которым относятся терминалы, принтеры и сети. Блочные специальные файлы используются для моделирования дисков.

Операции над файлами:

- Создание (create)
- Удаление (delete)
- Чтение (read) – последовательное и произвольное
- Открытие (open)
- Запись (write)
- Добавление (add)

Метаинформация - структурированные данные, представляющие собой характеристики описываемых сущностей для целей их идентификации, поиска, оценки, управления ими.

Атрибуты файла - метаданные: это каталог, только чтение, системный, скрытый, архивный.

Каталог — список объектов файловой системы (файлов и подкаталогов) с указанием их месторасположения на устройстве хранения информации (обычно на жёстком диске или компакт-диске) и прочими метаданными.

Структура каталога – одноуровневая и иерархическая.

Отсюда понятия: Полное имя файла(путь от корневого каталога) и относительное (короткое, относительно текущего каталога).

Файловая система - это механизм организации и хранения файлов на каком-либо носителе.

Задачи файловой системы

- Поиск секторов файла на диске по имени файла (уметь определить номера секторов)
- управление секторами на диске (уметь находить свободные сектора)

19. Основные структуры файловых систем

(определение из 18)

Файловая система - это часть общей системы управления памятью, назначение которой сводится в основном к управлению файлами, хранящимися во внешней памяти, а также к контролируемому разделению информации между пользователями.

Функции файловой системы

- предоставление возможности создавать, модифицировать, уничтожать файлы;
- контролируемое разделение файлов несколькими пользователями;
- предоставление пользователю возможности задания различной структуры файлов и возможности управления передачей информации между файлами;
- в системе должны быть предусмотрены средства обеспечения сохранности и восстановления информации в файлах;
- система должна обеспечивать независимость файлов от внешних устройств;
- система должна предоставлять защиту информации в файлах от несанкционированнного доступа (возможность шифрования и дешифрования данных);
- файловая система должна иметь "дружественный" интерфейс по отношению к пользователю.

Состав файловой системы

Файловая система, входящая в состав ядра ОС, как правило, содержит следующие средства:

- Методы доступа к хранящимся в файлах.
- Средства управления файлами
- Средства управления внешней памятью, обеспечивающие распределение пространства внешней памяти для размещения файлов.
- Средства обеспечения целостности файлов, которые гарантируют сохранность информации файла.

Примерная структура:

Суперблок	Структуры данных, описывающие свободное дисковое пространство и свободные индексные узлы	Массив индексных узлов	Блоки диска данных файлов

В начале раздела находится **суперблок**, содержащий общее описание файловой системы, например:

- Тип файловой системы
- Размер файловой системы в блоках
- Размер массива индексных узлов
- Размер логического блока и т. д.

Описанные **структуры данных** создаются на диске в результате его форматирования, их наличие позволяет обращаться к данным на диске как к файловой системе, а не как к обычной последовательности блоков.

В файловых системах современных ОС для повышения устойчивости поддерживается несколько копий суперблока.

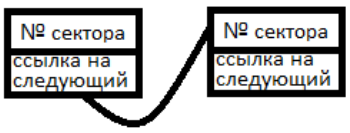
Массив индексных узлов (ilist) содержит список индексов, соответствующих файлам данной файловой системы. Размер массива индексных узлов определяется администратором при установке системы. Максимальное число файлов, которые могут быть созданы в файловой системе, определяется числом доступных индексных узлов.

В блоках данных хранятся реальные данные файлов. Размер логического блока данных может задаваться при форматировании файловой системы. Отдельно взятый блок данных может принадлежать одному и только одному файлу в файловой системе.

Основные ФС: -Windows: FAT и NTFS.
-Linux: Ext 2,3,4

FAT: Задача поиска секторов решается при помощи битовых карт. Как хранить списки секторов? Массив – плохо, т.к. в итоге диск получается фрагментированным.

Исп-ся динамический список:



Проблемы Fat: Потерянные кластеры (кластеры помечаются как исп-е, но не явл-ся таковыми; появляются при неверном завершении работы приложения или крахе системы. Решение: программы восстановления диска сохраняют инф-ю из таких кластеров и обнуляют их). Пересекающиеся файлы (появл-ся, когда две записи каталога указывают на один кластер, т.е. кластер содержит данные 2-х файлов, обычно один из файлов поврежден. Решение: программы восстановления диска копируют оба файла в свободное пр-во диска, а пересекающуюся область очищают). Chkdsk и fsk – проверка диска в W. и U.

20. Идентификация пользователей и права доступа

Идентификация пользователя позволяет субъекту назвать себя (избавление от анонимности – имя пользователя). Посредством аутентификации вторая сторона убеждается, что субъект действительно тот, за кого он себя выдает (происходит проверка подлинности). Предоставление определённому лицу или группе лиц прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий называется авторизацией.

Как проверить пользователя?

- Он что-то знает - имя, пароль (основной)
- Он что-то имеет - использование специальных устройств (магнитная карта и т.д.)
- Он кем-то является - использование биологической информации о пользователе (отпечатки пальцев, голос, главное в этой группе – сетчатка глаза)

Пароли не хранятся в открытом виде:
1) Хранение хэша=пароль + соль (добавка, известная серверу); 2) Одноразовые пароли (SMS-подтверждение); 3) Отклик-отзыв (при входе в систему задается один из вопросов из списка вопросов-ответов, который хранится на сервере, зашифрован.)

Права пользователя (права доступа к файлам):

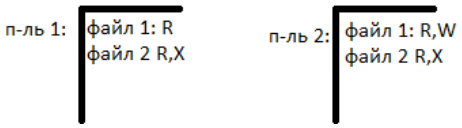
- чтение
- изменение
- удаление
- исполнение

Для хранения прав доступа к файлам используются таблица (матрица доступа): строки – субъекты, столбцы – объекты.

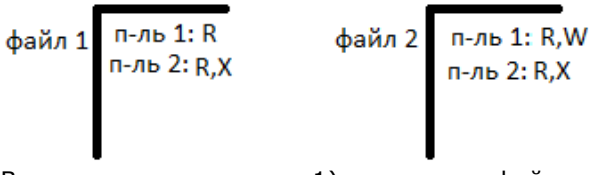
об. / суб.	файл 1	файл 2	и т.д.
польз-ль 1	R		
польз-ль 2	R,W	R,X	

Права: R (чтение), W(изменение), X (выполнение)
Таблицы могут быть:

- Таблицы по пользователю - c-list (разрезание по строкам)



- По файлу – ACL (Access Control list) (разрезание по колонкам).



Варианты оптимизации: 1) если много файлов с одинаковыми правами, то хранить ссылки на списки прав; 2) Использование групп пользователей (пользователи с одинаковыми правами)

Поскольку пользователей гораздо меньше, чем файлов, легче хранить права для каждого пользователя с самим файлом. Такая структура используется в большинстве ОС.

21. Виды атак на ОС.

Внутренние(инсайдерские):

1. Логическая бомба – это фрагмент кода, внедренный в основную программу (техническую систему и т.п), которая продолжает действовать нормально до определенного условия (пока вводится определенный пароль, проверка ведомости и т.п). Взрывы: стирание важных док-в, очистка диска и т.п
2. Лазейка – механизм, при котором можно обойти какую-либо проверку (например, войти в систему, обойдя все пароли).
3. Фальсификация входа в систему – программа, эмулирующая вход в систему – позволяет украсть логин и пароль

Использование дефектов кода (дефекты в коде ОС, внедрение туда своего кода):
1. Использующие переполнение буфера (уязвимость языка С в плане переполнения буфера, изменение имени файла – слишком длинное, в него внедрен код. Если с правами администратора – легко превратить машину в «зомби» - например, будет ожидать команд от опр. IP-порта)
2. Использующие форматирующую строку (printf в С – вывод строки формат способен перезаписать данные в памяти и передать управление в другое место)
3. Исп-е возвращение управления libc (Суть атаки в том, чтобы изловчиться и использовать strcpy для копирования программы атакующего, которую часто называют кодом оболочки, или шелл-кодом (shellcode), в сегмент данных и дать ей возможность выполниться в этом сегменте.)
4. Исп-е переполнения целочисленных знач-й (Возможность вызывать неопределяемые числовые переполнения может быть превращена в атаку. Один из способов атаки заключается в предоставлении программе двух допустимых (но больших) параметров, заранее зная, что над ними будет произведена операция сложения или умножения, в результате чего произойдет переполнение. Опр-е размера окна-> переполнение-> вызов распределителя malloc->атака с переполнением буфера)
5. Исп-е внедрения программного кода (Исп-е в коде system для вып-я команд + возм-ть получать имена файлов с клавиатуры=>поле для аттак: команда ср abc xyz: rm -rf / которая сначала копирует файл, а затем пытается рекурсивно удалить каждый файл и каждый каталог во всей файловой системе.)
6. Исп-е эскалации привилегий (для получения прав доступа – пример, cron-демон)

Внешние(вредоносное ПО):

1. Троянские кони – реальна программа с встроенным вредоносным модулем (игра и т.п.) При запуск программы, вредоносный модуль записывается на диск – и делает свое дело (удаляет файлы, отлавливает номера кредиток, пароли и т.п., ждет команд от I3-порта)
2. Вирусы – это программа, способная размножаться, присоединяя свой код к другим программам, Например: Записываются в .exe (вирусы, живущие в исполняемых файлах), заменяют точку входа. Умеют переписывать себя в другие программы. Макровирусы (живут в программах где есть макросы) – записываются в документ. Пишут макросы и что-нибудь делают.
3. Черви – это компьютерный код, распространяющийся без вмешательства пользователя. Жизненный цикл червей можно разделить на определенные стадии:

- Проникновение в систему
- Активация
- Поиск «жертв»
- Подготовка копий
- Распространение копий

Они живут в каналах связи, цель – найти IP. Червь-извозчик переносит Троян или Вирусы. Черви забивают сеть, трафик падает. Иногда понижается работоспособность пк.

Еще атаки:
DOS-атаки – это атаки, результатом которых является приведение атакуемой системы в нестабильное или полностью нерабочее состояние. Последствиями атак такого типа могут стать повреждение или разрушение информационных ресурсов, на которые они направлены, и, следовательно, невозможность их использования. Существует два основных типа DoS-атак:

- отправка компьютеру-жертве специально сформированных пакетов, не ожидаемых этим компьютером, что приводит к перезагрузке или остановке системы;
- отправка компьютеру-жертве большого количества пакетов в единицу времени, которые этот компьютер не в состоянии обработать, что приводит к исчерпанию ресурсов системы.

DDOS-атаки, тоже самое но с разных IP адресов (множественные атаки).

22. Иерархия классов безопасных ОС

Самым известным документом – стандартом безопасности компьютерных систем является «Критерии безопасности компьютерных систем», выпущенный в 1983 году, носит название «Оранжевая книга». Он описывает семь классов защищенности компьютерных систем.
Класс D. Минимальный уровень безопасности. В этот класс попадают системы, которые были заявлены на сертификацию, но ее не прошли. Все всё могут. (DOS)
Класс C1. Избирательная защита доступа. Среда класса C1 предназначена для пользователей, обрабатывающих данные одного уровня секретности. Предусматривает наличие достоверной вычислительной базы (TCB), выполнение требований к избирательной безопасности. Обеспечивается отделение пользователей от данных (меры по предотвращению считывания или разрушения данных, возможность защиты приватных данных). В настоящее время по этому классу сертификация не предусмотрена.
Защита, удовлетворяющая требованиям избирательной политики управления доступом, которая заключается в том, что для каждого объекта и субъекта в системе явно и недвусмысленно задается перечень допустимых типов доступа. Политика обеспечения безопасности - дискреционная. Идентификация и аутентификация: TCB (Trusted Computing Base - совокупность аппаратных и программных механизмов защиты, которые отвечают за поддержку политики безопасности) должна требовать, чтобы пользователи идентифицировали себя перед началом каких-либо действий, в которых TCB предполагает быть посредником. TCB обязательно использует один из механизмов аутентификации (например, пароли) для проверки подлинности идентификации пользователей.

Класс C2.

Управляемая защита доступа. Системы данного класса способны осуществлять более четко выделенный контроль в плане избирательной защиты доступа. Действия пользователя связываются с процедурами идентификации/аутентификации. Наделение и лишение пользователей привилегий доступа. Кроме этого, ведется аудит событий, критичных с точки зрения безопасности, выполняется изоляция ресурсов. Защита, основанная на управляемом доступе. К требованиям класса C1 добавляются требования уникальной идентификации субъекта доступа (любой пользователь должен иметь уникальное имя), защиты по умолчанию ("запрещено все, что не разрешено") и регистрации событий. В системах этого класса обязательно ведение системного журнала, где отмечаются события, связанные с безопасностью системы. Данные журнала должны быть защищены от доступа любых пользователей, за исключением администратора системы.

Класс B. Помимо вышеприведенных требований, системы класса B характеризуются полномочной моделью управления доступом, когда каждый субъект и объект системы снабжается метками конфиденциальности и решение на доступ принимается на основе сопоставления обеих меток. (Принудительная безопасность - многоуровневая)

B1. Меточная защита. Метки безопасности должны быть присвоены всем субъектам и объектам системы, которые могут содержать или получать конфиденциальную информацию. При этом должно контролироваться соответствие меток на данных, экспортируемых из системы с устройствами, на которые осуществляется вывод. Метка безопасности на вводимые данные запрашивается у пользователя.

B2. Структурированная защита. В этом классе система TCB должна опираться на четко определенную и документированную формальную модель политики безопасности. Действие избирательного и принудительного управления доступом распространяется на все субъекты и объекты в системе. Выявляются тайные каналы. TCB должна четко декомпозироваться на элементы, критичные и некритичные с точки зрения безопасности. Усиливаются механизмы аутентификации. Обеспечивается управление механизмами достоверности в виде поддержки функций системного администратора и оператора. Подразумевается наличие механизмов строгого управления конфигурацией.

B3. Домены безопасности. В системах этого класса в оборудовании должна быть реализована концепция монитора обращений, который должен:
– контролировать все взаимодействия субъектов с объектами;
– быть гарантированно защищен от несанкционированных изменений, порчи и подделки;
– быть простым для анализа и тестирования на предмет правильности выполнения обработки обращений (полнота тестов должна быть доказана).

Из системы защиты должен быть исключен код, который не требуется для обеспечения поддержки политики безопасности. Механизмы регистрации событий безопасности должны сразу же оповещать администратора и пользователей о нарушениях. Обязательным также является наличие процедур, обеспечивающих восстановление работоспособности системы.

Класс A. Верифицированное проектирование. Данный класс систем функционально эквивалентен классу B3 в том смысле, что не требуется добавления дополнительных архитектурных особенностей или предъявления иных требований к политике безопасности. Существенное отличие состоит в требовании наличия формальной спецификации проектирования и соответствующих методов верификации. В данном классе не зарегистрировано ни одной ОС.

23. Многоуровневые системы

Многоуровневая безопасность состоит в наличии разных уровней секретности, при этом на каждом уровне происходит сужение круга лиц. У каждого пользователя есть ранг, чем он выше, тем прав больше. У каждого файла соотв. есть уровень допуска.
Подход заключается в том, что все объекты могут иметь уровни секретности, а все субъекты делятся на группы, образующие иерархию в соответствии с уровнем допуска к информации. Существует две модели реализации:
1) Модель Белла Ла Падулы. Суть:

- Простое свойство секретности. Субъект может читать информацию только из объекта, уровень

секретности которого не выше уровня секретности субъекта. Генерал читает документы лейтенанта, но не наоборот.

- *-свойство. Субъект может записывать информацию в объекты только своего уровня или более высоких уровней секретности. Генерал не может случайно разгласить нижним чинам секретную информацию.

Отметим, что данная модель разработана для хранения секретов, но не гарантирует целостности данных. Например, здесь лейтенант имеет право писать в файлы генерала.

2) Модель Биба. Суть:

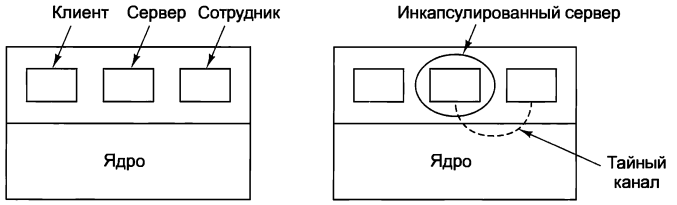
- Простой принцип целостности: процесс, работающий на уровне безопасности k, может осуществлять запись только в объекты своего или более низкого уровня (никакой записи наверх).
- Свойство целостности *: процесс, работающий на уровне безопасности k, может осуществлять чтение из объектов своего или более высокого уровня (никакого чтения из нижних уровней).

Гарантирует целостность данных, но не гарантирует безопасность.

В идеале: сочетать 2 метода, но т.к. они противоречивы, реализация сложна.

24. Проблема тайных ходов в операционной системе

Модель Лэмпсона: Идея заключается в том, что даже при надлежащем уровне безопасности может произойти утечка информации.



9.12. Клиентский, серверный и сотрудничающий процессы (*а*); вполне возможная утечка информации от инкапсулированного сервера к сотруднику через тайные каналы (*б*)

В чистейшем виде в модели задействованы три процесса на некой защищенной машине**. Первый процесс представлен клиентом**, желающим, чтобы **вторым процессом, сервером**, было выполнено некоторое задание. Клиент и сервер не вызывают друг у друга доверия. Третий процесс является сотрудником, вступающим в сговор с сервером именно для того, чтобы похитить конфиденциальные данные клиента.

Проблема ограждения – создать систему, предотвращаю утечку данных клиента из сервера, которые получены вполне законно. Решение: инкапсулировать сервер, чтобы не смог передать инф-ю сотруднику (матрицы защиты).

сервер может передавать следующий поток двоичных битов. Но сущ-ют способы утечки, менее заметные. Сервер передает посл-ть бит. Для отправки единичного бита он может заниматься вычислением на полную мощность в течение определенного интервала времени, а для передачи нулевого бита он на тот же период времени может прекращать вычислительный процесс. Сотрудник может попытаться обнаружить поток битов за счет тщательного отслеживания времени отклика этого процесса. Обычно он будет получать более быстрый отклик, когда сервер отправляет 0, чем тогда, когда он отправляет 1**. Этот канал связи называется тайным каналом (covert channel).**

Один из видов тайных каналов, это стеганография – передача информации так, что ее не видно. Каков принцип работы этого тайного канала? Оригинальное изображение состоит, например, из 1024 x 768 пикселов. Каждый пиксел состоит из трех 8-разрядных чисел, по одному для кодирования яркости красной, зеленой и синей составляющей пиксела. Цвет пикселов формируется за счет линейного наложения трех цветов. Метод кодирования использует в качестве тайного канала младший разряд каждого цветового значения RGB. Таким образом, в каждом пикселе появляется пространство для трех битов секретной информации, по одному в красной, зеленой и синей составляющей. При таких размерах изображения в нем может быть сохранено до 1024 x 768 x 3 бита, или 294 912 байтов секретной информации. (При этом картинка не меняется.)

Еще один вид, криптография – шифрование.

Идея: существует сообщение или файл – передать в зашифрованном виде.

1) Шифрование с секретным ключом (есть тайный ключ, например, последов-ть букв, в которую переименуется каждая буква алфавита, его должны знать и отправитель, и получатель.)
2) Шифр-е с открытым ключом (Способ работы заключается в том, что все получают пару (открытый ключ, закрытый ключ), а открытый ключ публикуется. Открытый ключ является ключом шифрования, закрытый — дешифрования. Обычно генерация ключей происходит в автоматическом режиме, возможно, с использованием выбранного пользователем пароля в качестве передаваемого алгоритму начального числа. Для отправки пользователю секретного сообщения корреспондент зашифровывает текст этого сообщения открытым ключом получателя. Поскольку закрытый ключ есть только у получателя, только он в состоянии расшифровать сообщение.)
3) Односторонние функции (спец. функции, такие, что y=f(x) – дается x, получаем y, но только по f(x) нельзя получить x) – криптографические хэш-ф-ии. Обычно исп-ся для цифровой подписи.

Криптографический процессор – место для хранения ключей