1. Процедура загрузки ОС в оперативную память.

Загрузка ОС(идея):

- Сначала идет тестирование системы. Система пытается определить что подключено к системе.
- Загружается программа начальной загрузки BIOS. Её функция сформировать часть таблицы векторов прерывания и переписать программу для работы с диском.
- Бутовый загрузчик (находится в MBR). По нему определяется активный раздел и в соответствии с инф. в MBR находит дорожку, сектор и т.д., где находится начало.
- Загрузка ОС: обычно 2 основные программы инициализации:
  - Инициализация ядра. В виду того, что ядро ОС представляет собой совокупность взаимосвязанных модулей, имеющих связь друг с другом, процесс структурной организации модулей называется инициализацией. Инициализация системы - процесс создания ядра системы который включает не только перезапись программ, но и системных структур, обеспечивающих знания системы о ее параметрах. Программа - загрузчик.
  - Инициализация системы. При загрузке ОС необходимо выполнить связывание, размещение всех частей, входящих в ядро супервизора, т.е. размещение резидентных программ на своих местах, формирование специальных системных структур данных в области данных операционной системы, формирование постоянной области, активизацию процессов для начала работы операционной системы. Этот процесс называется инициализацией операционной системы.

Загрузчик BIOS

Выполняет:

- инициализацию основных компонентов материнской платы;
- обслуживает системные прерывания (как аппаратные, так и программные);
- из Main Boot Record считывает первые 512 байт (бутовый/начальный загрузчик) в ОП. Передает ему управление.

Бутовый (первичный) загрузчик

Определяет активный раздел. Обращается к месту на жестком диске где записан основной загрузчик (обычно к 0 разделу 0 дорожки) и загружает основной загрузчик в память.

Основной (вторичный) загрузчик

Основной загрузчик системы инициализирует некоторые из подсистем (система ввода/вывода => файл конфигурации). Он может загружать несколько операционных систем, давая пользователю выбирать, какую из систем нужно загружать в конкретном случае. После выбора загружаемой системы, загрузчик проводит необходимые приготовления (к примеру, переводит процессор в защищенный режим работы) и начинает загрузку частей ядра в ОП (программы обработки прерываний, управление памятью, управление процессами). После этого загрузчик передает управление ядру (при этом возможна передача параметров. К примеру, при загрузке Linux ядру можно передавать настройки графического и режима и другие параметры).

После формирования ядра начинает работу программа инициализации системы. Подгружается командный интерпретатор (он грузится последним потому, что мы можем указать свой собственный интерпретатор).

2. PCB и его роль в управлении процессами. Структура. Назначение основных полей.

Выполнение функций ОС, связанных с управлением процессами, осуществляется с помощью специальных структур данных, образующих окружение процесса, среду исполнения или образ процесса. Образ процесса состоит из двух частей: данных режима задачи и режима ядра. Образ процесса в режиме задачи состоит из сегмента кода программы, которая подчинена процессу, данных, стека, библиотек и других структур данных, к которым он может получить непосредственный доступ. Образ процесса в режиме ядра состоит из структур данных, недоступных процессу в режиме задачи, которые используются ядром для управления процессом. Оно содержит различную вспомогательную информацию, необходимую ядру во время работы процесса. Каждому процессу в ядре операционной системы соответствует блок управления процессом (PCB – process control block). Вход в процесс (фиксация системой процесса) –это создание его блока управления (PCB), а выход из процесса – это его уничтожение, т. е. уничтожение его блока управления.

Таким образом, для каждой активизированной задачи система создает свой PCB, в котором, в сжатом виде, содержится используемая при управлении информация о процессе.

PCB – это системная структура данных, содержащая определённые сведения о процессе со следующими полями:

1. Идентификатор процесса (имя);

2. Идентификатор родительского процесса;
3. Текущее состояние процесса (выполнение, приостановлен, сон и т.д.);
4. Приоритет процесса;
5. Флаги, определяющие дополнительную информацию о состоянии процесса;
6. Список сигналов, ожидающих доставки;
7. Список областей памяти выделенной программе, подчиненной данному процессу;
8. Указатели на описание выделенных ему ресурсов;
9. Область сохранения регистров;
10. Права процесса (список разрешенных операций);

3. Основные характеристики распределенных систем обработки информации

Распределённые вычислительные системы

Среда, в которой компоненты системы или ресурсы: процессоры, память, принтеры, графические станции, программы, данные и т.д., связанны вместе посредством сети, которая позволяет пользователям представлять ВС как единую вычислительную среду и иметь доступ к ее ресурсам.

Особенности:

- **Ограниченность** связана с тем, что количество узлов в сети ограничено и они являются независимыми компонентами сети.
- **Идентификация**. Каждый из ресурсов сети должен однозначно идентифицироваться (именоваться, например).
- **Распределенное управление**. Каждый узел должен иметь возможность и средства (hardware, software) для управления сетью. Но с т.з. надёжности нежелательно давать каким-то узлам больше привилегий в управлении.
- **Гетерогенность**. Система должна работать на разнородных узлах с различной длиной слов и байтовой организацией. Это касается программного, прикладного и системного обеспечения узлов.

Пользователь не должен знать особенности аппаратного обеспечения сети.

Распределённые операционные системы

РОС строятся на РВС. **Свойства** распределенных операционных систем:

- Надо поддерживать когерентность файлов.
- Распределенная система распределяет выполняемые работы в узлах системы, исходя из соображений повышения пропускной способности всей системы;
- Распределенные системы имеют высокий уровень организации параллельных вычислений

Принципы построения распределенных ОС

1) **Прозрачность** (для пользователя и программы). Прозрачность сети требует, чтобы детали сети были скрыты от конечных пользователей.
   - расположения – пользователь не должен знать, где расположены ресурсы
   - миграции – ресурсы могут перемещаться без изменения их имен
   - размножения – пользователь не должен знать, сколько копий существует
   - конкуренции – множество пользователей разделяет ресурсы автоматически
   - параллелизма – работа может выполняться параллельно без участия пользователя
   - именование – имя должно быть уникальным в глобальном смысле, и не имеет значения в каком месте системы оно будет использовано
2) **Гибкость** (не все еще ясно - потребуется менять решения). Использование монолитного ядра ОС или микроядра.
3) **Надежность.** Доступность, устойчивость к ошибкам (fault tolerance).Секретность.
4) **Производительность**. Грануллированность. Мелкозернистый и крупнозернистый параллелизм (fine-grained parallelism, coarse-grained parallelism). Устойчивость к ошибкам требует дополнительных накладных расходов.
5) **Масштабируемость** – система может подключать дополнительное оборудование для увеличения производительности.

Плохие решения:

- централизованные компоненты (один почтовый сервер);
- централизованные таблицы (один телефонный справочник);
- централизованные алгоритмы (маршрутизатор на основе полной информации).

Только децентрализованные алгоритмы со следующими чертами:

- ни одна машина не имеет полной информации о состоянии системы;
- машины принимают решения на основе только локальной информации;
- выход из строя одной машины не должен приводить к отказу алгоритма;
- не должно быть неявного предположения о существовании глобальных часов.

Отличие распределённой ОС от сетевой

В сетевой операционной системе пользователи знают о существовании многочисленных компьютеров, могут регистрироваться на удаленных машинах и копировать файлы с одной машины на другую. Каждый компьютер работает под управлением локальной операционной системы и имеет своего собственного локального пользователя. Сетевые операционные системы несущественно отличаются от однопроцессорных операционных систем. Ясно, что они нуждаются в сетевом интерфейсном контроллере и специальном низкоуровневом программном обеспечении, поддерживающем

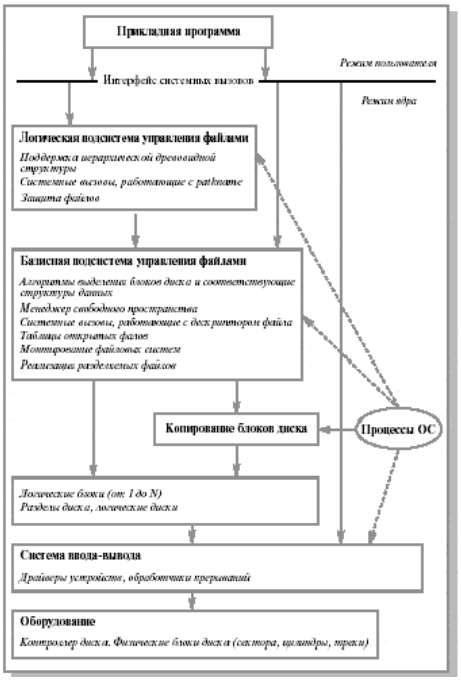
работу контроллера, а также в программах, разрешающих пользователям удаленную регистрацию в системе и доступ к удаленным файлам.
Распределенная операционная система, напротив, представляется пользователям традиционной однопроцессорной системой, хотя она составлена из множества процессоров. При этом пользователи не должны беспокоиться о том, где работают их программы или расположены файлы; все это должно автоматически и эффективно обрабатываться самой ОС.

4. Взаимодействие файловой системы с другими частями ОС.

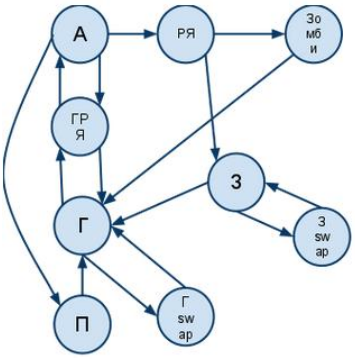
Разработчикам файловых систем приходится заботиться о том, как файлам выделяется место на диске, и о том, как система следит за тем, какой блок какому файлу принад лежит. Различные варианты реализации файлов включают в себя непрерывные файлы, связные списки, таблицы размещения файлов и i-узлы. В различных сис темах используются различные каталоговые структуры. Атрибуты файла могут храниться прямо в каталоге или в другом месте (например, в i-узле). Учет дискового пространства может осуществляться с помощью списков свободных блоков или битовых массивов. Надежность файловых систем может быть увеличена при помощи создания инкрементных резервных копий, а также с помощью программы, способной исправлять поврежденные файловые системы. Производительность файловых систем также является важным вопросом. Она может быть увеличена различными способами, включая кэширование, опережающее чтение и размещение блоков файла рядом друг с другом. Файловые системы с журнальной структурой тоже увеличивают производительность, выполняя операции записи большими блоками данных.
Файловая система - это часть операционной системы, назначение которой состоит в том, чтобы организовать эффективную работу с данными, хранящимися во внешней памяти, и обеспечить пользователю удобный интерфейс при работе с такими данными. Организовать хранение информации на магнитном диске непросто. Это требует, например, хорошего знания устройства контроллера диска, особенностей работы с его регистрами. Непосредственное взаимодействие с диском - прерогатива компонента системы ввода-вывода ОС, называемого драйвером диска. Для того чтобы избавить пользователя компьютера от сложностей взаимодействия с аппаратурой, была придумана ясная абстрактная модель файловой системы. Операции записи или чтения файла концептуально проще, чем низкоуровневые операции работы с устройствами.
Основная идея использования внешней памяти состоит в следующем. ОС делит память на блоки фиксированного размера, например, 4096 байт. Файл, обычно представляющий собой неструктурированную последовательность однобайтовых записей, хранится в виде последовательности блоков (не обязательно смежных); каждый блок хранит целое число записей. В некоторых ОС (MS-DOS) адреса блоков, содержащих данные файла, могут быть организованы в связный список и вынесены в отдельную таблицу в памяти. В других ОС (Unix) адреса блоков данных файла хранятся в отдельном блоке внешней памяти (так называемом индексе или индексном узле). Этот прием, называемый индексацией, является наиболее распространенным для приложений, требующих произвольного доступа к записям файлов. Индекс файла состоит из списка элементов, каждый из которых содержит номер блока в файлеи сведения о местоположении данного блока. Считывание очередного байта осуществляется с так называемой текущей позиции, которая характеризуется смещением от начала файла. Зная размер блока, легко вычислить номер блока, содержащего текущую позицию. Адрес же нужного блока диска можно затем извлечь из индекса файла. Базовой операцией, выполняемой по отношению к файлу, является чтение блока с диска и перенос его в буфер, находящийся в основной памяти.
Файловая система позволяет при помощи системы справочников (каталогов, директорий) связать уникальное имя файла с блоками вторичной памяти, содержащими данные файла. Иерархическая структура каталогов, используемая для управления файлами, может служить другим примером индексной структуры. В этом случае каталоги или папки играют роль индексов, каждый из которых содержит ссылки на свои подкаталоги. С этой точки зрения вся файловая система компьютера представляет собой большой индексированный файл. Помимо собственно файлов и структур данных, используемых для управления файлами (каталоги, дескрипторы файлов, различные таблицы распределения внешней памяти), понятие "файловая система" включает программные средства, реализующие различные операции над файлами.

Перечислим основные функции файловой системы:

1. Идентификация *файлов*. Связывание имени *файла* с выделенным ему пространством *внешней памяти*.
2. Распределение *внешней памяти* между *файлами*. Для работы с конкретным *файлом* пользователю не требуется иметь информацию о местоположении этого *файла* на внешнем носителе информации. Например, для того чтобы загрузить документ в редактор с жесткого диска, нам не нужно знать, на какой стороне какого магнитного диска, на каком цилиндре и в каком секторе находится данный документ.
3. Обеспечение надежности и отказоустойчивости. Стоимость информации может во много раз превышать стоимость компьютера.
4. Обеспечение защиты от несанкционированного доступа.
5. Обеспечение совместного *доступа к файлам*, так чтобы пользователю не приходилось прилагать специальных усилий по обеспечению синхронизации доступа.
6. Обеспечение высокой производительности.



5. Состояния процесса и особенности перехода из одного состояния в другое.



П – подготовка. На этом этапе лежат задания. Т.е. программа (что делать) и данные (над чем делать). Заданию ещё не выделили ресурсы. Когда ему планировщик выделит ресурсы, задание станет процессов и перейдёт в готовое состояние.

Г – готовность. Процесс размещен в ОП, ему выделены ресурсы, сформирован РСВ. Так может случиться, что процесс своппируют на диск вместе с его ресурсами (ну, разве что ОП из ресурсов вычеркнут).

Гswap – процесс готов, но своппирован на диске.

ГРя – готовность в режиме ядра. Это некое абстрактное состояние, когда процесс уже имеет ресурсы, но ещё не допущен к процессору. С этого состояния процесс может или получить долгожданный доступ к процессору, или вернуться в очередь готовых.

А – активность. Процесс занял процессор и, собственно, выполняется. Если у процесса окончились выделенные ему кванты времени, он возвращается в ГРя. Если внезапно окончились его ресурсы – в очередь подготовленных. Если произошёл системный вызов – в Ря.

Ря – режим ядра. Сюда попадает процесс, пока выполняется системный вызов (например, ядро открывает файл). Процесс может поступить к заблокированным, или стать зомби.

Зомби – процесса в системе нет, но его РСВ ещё есть. Такая ситуация возникает, когда процесс завершился, а породивший его процесс ещё не знает об этом.

Заблокированные процессы находятся в состоянии ожидания. Кому не повезёт – освободят ОП и будут своппированны на диск.

Зswap – процесс с диска можно опять вернуть в ОП в очередь заблокированных.

Если система определила необходимость активизации процесса и выделяет нужные ему ресурсы, кроме времени процессора, то она переводит его в готовое состояние.

Если процессор освободился, то первый (наиболее приоритетный) процесс из очереди готовых процессов получает время процессора и переходит в активное состояние. Выделение времени процессора процессу осуществляет диспетчер. В состоянии исполнение происходит непосредственное выполнение программного кода процесса. Выйти из этого состояния процесс может по трем причинам:

- операционная система прекращает его деятельность;
- он не может продолжать свою работу, пока не произойдет некоторое событие, и операционная система переводит его в состояние ожидание;
- в результате возникновения прерывания в вычислительной системе (например, прерывания от таймера по истечении предусмотренного времени выполнения) его возвращают в состояние готовность
- процесс не выполнился за выделенный ему квант времени, тогда он переходит снова в готовое состояние.

Из состояния ожидание процесс попадает в состояние готовность после того, как ожидаемое событие произошло, и он снова может быть выбран для исполнения.

Если процесс требует действия или ресурса, которых в данный момент операционная система не может выполнить, он переводится в подготовленное состояние и считается приостановленным.

В общем случае, система должна следить за процессами в очередях готовых и заблокированных процессов, чтобы не было бесконечно ожидающих процессов или процессов, монопольно удерживающих выделенные им ресурсы.