ЛЕКЦИЯ №2

«Концептуальные основы операционных систем»

по дисциплине

«Безопасность операционных систем»

Текст лекции рассмотрен и одобрен на заседании кафедры протокол № _____ от " " 201__ г.

(Слайд 1. Титульный слайд)

Уважаемые студенты! Сегодня вы продолжаете изучение дисциплины «Безопасность операционных систем». Лекция №2 «Концептуальные основы операционных систем». Продолжительность лекции - 4 академических часа.

Слайд 2 (план проведения занятия)

Данная лекция состоит из 3-х логических частей. Первая часть — введение. Во введении вы узнаете о задачах и содержании настоящей дисциплины, программу дисциплины, вам будет рекомендована литература для изучения и закрепления материала.

Вторая часть — основная. В ней мы вспомним основные компоненты ОС. Ядро и микроядро. Управление памятью. Межпроцессное взаимодействие. Управление вводом-выводом. Управление задачами. Пользовательские интерфейсы в современных ОС. Виртуализация.

Третья часть — заключение. Будут даны рекомендации для самостоятельной работы студента, сделаны выводы по материалам лекции, выражены пожелания по индивидуальному развитию студентов как высококлассных IT-специалистов в области информационной безопасности.

Функции ОС

Основными функциями ОС являются:

- автоматическое выполнение действий по запуску задач в обработку и их завершению;
- диспетчеризация (планирование обработки задач);
- распределение памяти между различными задачами;
- управление ходом выполнения задач в вычислительной системе;
- распределение задачам необходимых ресурсов ВС;
- синхронизация выполнения задач;
- поддержка выполнения операций ввода/вывода данных;
- ведение учета работы системы (при необходимости).

Выполнение своих функций ОС осуществляется с помощью соответствующих программных комплексов управления, которые носят название супервизорных программ (супервизоров или менеджеров).

Принципы построения ОС

Частотный принцип реализации системных программ основан на выделении в алгоритмах и в обрабатываемых массивах ОС действий и данных по частоте их использования. Следствием применения частотного принципа в современных ОС – наличие многоуровневого планирования при организации работы ОС.

Принцип модульности отражает технологические и эксплуатационные свойства системы, предусматривая оформление функционально законченных компонентов ОС в виде отдельных модулей.

Принцип функциональной избирательности предусматривает выделение некоторого множества важных модулей, которые должны быть постоянно в "горячем" режиме для обеспечения эффективного управления вычислительным процессом. Этот выделенный набор модулей называют ***ядром ОС***. При формировании состава ядра ОС ищут компромисс между двумя разноречивыми требованиями: в состав ядра должны войти наиболее часто используемые модули; объем памяти, занимаемый ядром ОС, должен быть как можно меньше. Программы ядра ОС постоянно находятся в оперативной памяти ЭВМ и называются резидентными. Программы ОС, подгружаемы в ОЗУ по мере необходимости из внешней памяти, называются транзитными.

Принцип генерируемости определяет такой способ исходного представления системной программы ОС, который позволяет настраивать эту системную программу, исходя из конкретной конфигурации аппаратных средств и круга решаемых проблем.

Принцип функциональной избыточности предусматривает обеспечение возможности выполнения одной и той же работы различными средствами.

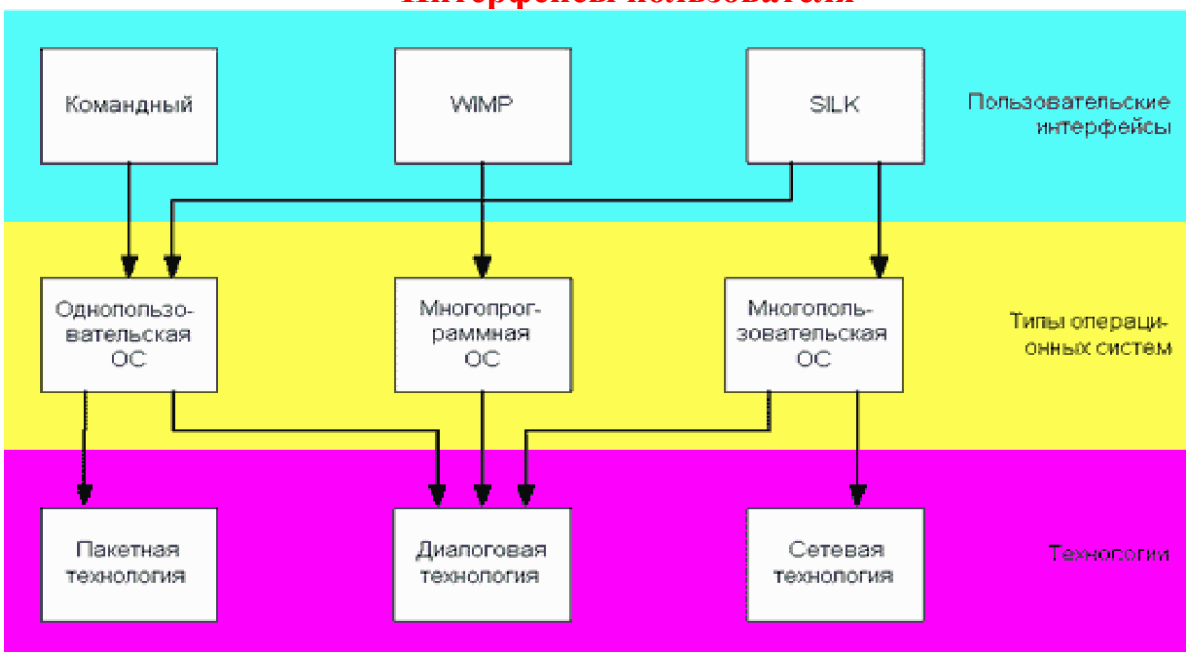
Принцип перемещаемости предусматривает такое построение модулей ОС, при котором результаты работы не зависят от места их расположения.

Принцип защиты информации определяет необходимость разработки мер, ограждающих программы и данные пользователя от искажений или нежелательных влияний друг от друга, а также пользователей на ОС и обратно.

Принцип независимости программ от внешних устройств заключается в том, что связь программ с конкретными внешними устройствами осуществляется не на уровне подготовки программных устройств (трансляции или компиляции исходного кода, генерации выполняемого модуля), а в период планирования операционной системой ее выполнения.

Принцип открытости и наращиваемости ОС предусматривает возможность доступа к ней для анализа пользователями, специалистами, обслуживающим персоналом, а также изменения конфигурации ОС и ее мощности без осуществления процессов генерации.

Интерфейсы пользователя



Современными видами интерфейсов являются:

- Командный интерфейс – называется так по тому, что в этом виде интерфейса человек подает "команды" компьютеру, а компьютер их выполняет и выдает результат человеку. Командный интерфейс

реализован в виде пакетной технологии и технологии командной строки.

- WIMP-интерфейс (Window – окно, Image – образ, Menu – меню, Pointer – указатель). Характерной особенностью этого вида интерфейса является то, что диалог с пользователем ведется не с помощью команд, а с помощью графических образов – меню, окон, других элементов. Хотя и в этом интерфейсе подаются команды машине, но это делается "опосредственно", через графические образы. Этот вид интерфейса реализован на двух уровнях технологий: простой графический интерфейс и "чистый" WIMP – интерфейс.
- SILK-интерфейс (Speech – речь, Image – образ, Language – язык, Knowlege – знание). Этот вид интерфейса наиболее приближен к обычной, человеческой форме общения. В рамках этого интерфейса идет обычный "разговор" человека и компьютера. При этом компьютер находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы. Результат выполнения команд он также преобразует в понятную человеку форму. Этот вид интерфейса наиболее требователен к аппаратным ресурсам компьютера, и поэтому его применяют в основном для военных целей.
- Общественный интерфейс – основан на семантических сетях.

Прерывания

Прерывание – временное прекращение процесса, такого как выполнение программы вычислительной машины, вызванное событием, внешним по отношению к этому процессу, и совершенное таким образом, что процесс может быть продолжен

В вычислительной машине прерывание – это событие, при котором меняется нормальная последовательность команд, выполняемых процессором. Сигнал "прерывание" сначала отрабатывается аппаратурой вычислительной машины – системой прерываний.
Если произошло прерывание, то в вычислительной системе выполняются последовательно следующие действия:

- управление передается операционной системе;
- операционная система запоминает состояние прерванного процесса;
- операционная система анализирует тип прерывания и передает управление соответствующей программе обработки этого прерывания;
- программа обработки прерывания выполняет предписанные действия и передает управление операционной системе;
- операционная система по результатам работы программы обработки прерываний либо восстанавливает состояние прерванного процесса и позволяет развиваться ему дальше, либо аварийно заканчивает его.

Следует иметь в виду, что инициатором прерывания может быть также и выполняющийся процесс.

Количество источников сигналов прерывания достигает в современных вычислительных системах нескольких сотен и даже тысяч. Все возможные в системе прерывания можно классифицировать по месту (причине) их возникновения.

Различают шесть основных классов прерываний:

- прерывания от схем контроля ЭВМ;
- прерывания по рестарту (повторному пуску);
- прерывания ввода/вывода;
- внешние прерывания;
- прерывания по вызову супервизора;
- программные прерывания.

- Прерывание от схем контроля возникает в случае появления любой аппаратной ошибки в ЭВМ. Продолжение работы машины становится невозможным, и процесс аварийно заканчивает свое существование.
- Прерывание по рестарту может наступить в следующих случаях: на пульте управления была нажата кнопка (клавиша, сочетание клавиш) повторного пуска ЭВМ; процесс, выполняющий в данной ЭВМ, выдал команду рестарта; в многомашинной системе получена команда рестарта от другого компьютера. В любом случае в ЭВМ, получившей команду рестарта, выполняются действия по загрузке операционной системы.
- Прерывания ввода/вывода инициируются аппаратурой, обеспечивающей операции ввода и вывода данных. Они сигнализируют центральному процессору об изменении состояния устройств ввода/вывода.
- Внешнее прерывание может возникнуть по самым различным причинам. Типичными источниками внешних прерываний являются таймер, выдающий сигнал об истечении кванта времени, параллельный синхронный или асинхронный процесс, другой процессор, другой компьютер.
- Прерывание по вызову супервизора появляется в том случае, когда работающий процесс выполняет команду обращения к супервизору. Этой командой (SVC-командой) программа пользователя генерирует запрос на предоставление конкретной системной услуги, например, на выполнение операции ввода/вывода, увеличение объема выделенной памяти и т.п. Механизм SVC-команд позволяет защитить операционную систему от пользовательских процессов.
- Программное прерывание может возникнуть по двум причинам: процесс пытается выполнить ошибочную операцию (например, деление на нуль, операция с неправильным кодом и т.п.); процесс выполнил заранее подготовленную команду прерывания для обеспечения перехода к выполнению других действий (например, для синхронизации нескольких процессов в вычислительной системе).

Ядро

Понятие ***ядра ОС*** непосредственно вытекает из принципов построения ОС, конкретно – из принципа избирательности. Принцип функциональной избирательности предусматривает выделение некоторого множества важных модулей, которые должны быть постоянно в "горячем" режиме для обеспечения эффективного управления вычислительным процессом. Этот выделенный набор модулей называют ядром ОС.

Микроядро

Ядро любой современной ОС представляет собой набор очень большого количества функций, с запутанными взаимосвязями и очень расплывчатыми границами между основными подсистемами. В результате любая модификация организованной таким образом системы дается тяжело и приводит к появлению в новых версиях большого количества ошибок. Кроме того, не во всех инсталляциях нужны все компоненты ядра, а при монолитном его построении удаление ненужных функций затруднено. Недостатки, присущие операционным системам с большим монолитным ядром (а это, в первую очередь, различные версии Unix), породили интерес к системам, построенным на основе микроядра.

Микроядерный подход заключается в том, что базовые функции ядра оформляются в виде отдельной небольшой компоненты, выполняемой в привилегированном режиме, а остальные функции ОС выполняются в пользовательском режиме с использованием примитивов микроядра.

Основной сложностью использования микроядерного подхода на практике является замедление скорости выполнения системных вызовов при передаче сообщений через микроядро по сравнению с классическим подходом.

МУЛЬТИЗАДАЧНОСТЬ, ПРОЦЕССЫ И НИТИ

Мультизадачность. В современном мультипрограммировании различают два типа мультизадачности: кооперативная и вытесняющая.

Кооперативная мультизадачность – это такой режим работы ОС, когда активный процесс, которому ОС выделило центральный процессор, монопольно использует его до тех пор, пока ему не потребуется выполнить какие-либо операции с внешними устройствами.

Вытесняющая мультизадачность – это такой режим работы ОС, когда операционная система в любой момент времени может приостановить развитие активного процесса, сохранив его состояние во внешней памяти ("вытеснить" процесс), и активизировать вместо него другой процесс.

Процессы. Важнейшей частью операционной системы, непосредственно влияющей на функционирование вычислительной машины, является подсистема управления процессами. **Процесс** (или по-другому, задача) – абстракция, описывающая выполняющуюся программу. Для операционной системы процесс представляет собой единицу работы, заявку на потребление системных ресурсов. Подсистема управления процессами планирует выполнение процессов, т.е. распределяет процессорное время между несколькими одновременно существующими в системе процессами, а также занимается созданием и уничтожением процессов, обеспечивает процессы необходимыми системными ресурсами, поддерживает взаимодействие между процессами.

Контекст и дескриптор процесса. На протяжении существования процесса его выполнение может быть многократно прервано и продолжено. Для того чтобы возобновить выполнение процесса, необходимо восстановить состояние его операционной среды. Состояние операционной среды отображается состоянием регистров и программного счетчика, режимом работы процессора, указателями на открытые файлы, информацией о незавершенных операциях ввода-вывода, кодами ошибок выполняемых данным процессом системных вызовов и т.д. Эта информация называется ***контекстом процесса***.

Кроме этого, операционной системе для реализации планирования процессов требуется дополнительная информация: идентификатор процесса, состояние процесса, данные о степени привилегированности процесса, место нахождения кодового сегмента и другая информация. В некоторых ОС (например в ОС Unix) информацию такого рода, используемую ОС для планирования процессов, называют ***дескриптором процесса***.

Дескриптор процесса по сравнению с контекстом содержит более оперативную информацию, которая должна быть легко доступна подсистеме планирования процессов. Контекст процесса содержит менее актуальную информацию и используется операционной системой только после того, как принято решение о возобновлении прерванного процесса.

Очереди процессов представляют собой дескрипторы отдельных процессов, объединенные в списки. Таким образом, каждый дескриптор, кроме всего прочего, содержит, по крайней мере, один указатель на другой дескриптор, соседствующий с ним в очереди. Такая организация очередей позволяет легко их переупорядочивать, включать и исключать процессы, переводить процессы из одного состояния в другое.

Программный код только тогда начнет выполняться, когда для него операционной системой будет создан процесс.

Создать процесс – это значит:

- создать информационные структуры, описывающие данный процесс, т.е. его дескриптор и контекст;
- включить дескриптор нового процесса в очередь готовых процессов;

- загрузить кодовый сегмент процесса в оперативную память или в область свопинга.

Нити. Многозадачность является важнейшим свойством ОС. Для поддержки этого свойства ОС определяет и оформляет для себя те внутренние единицы работы, между которыми и будет разделяться процессор и другие ресурсы компьютера. Эти внутренние единицы работы в разных ОС носят разные названия: задача, задание, процесс, нить. В некоторых случаях сущности, обозначаемые этими понятиями, принципиально отличаются друг от друга. Говоря о процессах, мы отмечали, что операционная система поддерживает их обособленность: у каждого процесса имеется свое виртуальное адресное пространство, каждому процессу назначаются свои ресурсы – файлы, окна, семафоры и т.д. Такая обособленность нужна для того, чтобы защитить один процесс от другого, поскольку они, совместно используя все ресурсы машины, конкурируют с друг другом. В общем случае процессы принадлежат разным пользователям, разделяющим один компьютер, и ОС берет на себя роль арбитра в спорах процессов за ресурсы.

Нити имеют собственные:

- программный счетчик;
- стек;
- регистры;
- нити-потомки;
- состояние.

Нити разделяют:

- адресное пространство;
- глобальные переменные;
- открытые файлы;
- таймеры;
- семафоры;
- статистическую информацию.

Память

В операционных системах различают два вида памяти: основная (первичная) и внешняя (вторичная).

Основная память (main storage) – оперативная память центрального процессора или ее часть, представляющая собой единое пространство памяти.

Внешняя память (external storage) – память, данные в которой доступны центральному процессору посредством операций ввода-вывода.

Для непосредственного выполнения программ или обращения к данным необходимо, чтобы они размещались в основной памяти. Внешняя память имеет, как правило, гораздо большую емкость, чем основная, стоит

дешевле и позволяет хранить данные и программы, которые должны быть наготове для обработки.

Кроме основной и внешней памяти в современных ЭВМ существует дополнительная быстродействующая память, называемая ***кэш-памятью***.

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

Существует два наиболее известных способа реализации виртуальной памяти – страничный и сегментный. Применяется также их комбинация – странично-сегментная организация виртуальной памяти.

В настоящее время применяются следующие стратегии выталкивания (откачки) страниц (сегментов):

- выталкивание случайных страниц или сегментов;
- выталкивание первой пришедшей страницы или сегмента (FIFO);
- выталкивание дольше всего не использовавшихся страниц или сегментов (LRU – Least Recently Used);
- выталкивание наименее часто использовавшихся страниц или сегментов (LFU – Least Frequently Used);
- выталкивание не использовавшихся в последнее время страниц или сегментов (NRU – Not Recently Used).

Данные

Данные – это информация, представленная в виде, пригодном для обработки автоматическими средствами при возможном участии человека.

Любые данные, подлежащие обработке, находятся на том или ином носителе.

Носитель данных – это материальный объект, предназначенный для хранения данных. Носителями данных могут служить магнитные диски и ленты, перфокарты и перфоленты, компакт-диски и т.п. Кроме того, данные могут поступать в систему обработки от источников данных, не являющихся средствами хранения информации.

Примерами такого рода источников данных являются линии передачи информации, аналого-цифровые преобразователи, к входам которых подключены выходы измерительных приборов и систем и т.п.

Источник данных – функциональное устройство, являющееся источником передаваемых данных.

Источник информации – часть коммуникационной системы, которая порождает сообщение.

Методы доступа к данным

Метод прямого управления доступом к данным основан на наличии непосредственной связи между центральным процессором и внешним запоминающим устройством.

Метод косвенного управления доступом основан на том, что между центральным процессором и внешними запоминающими устройствами помещается специальный процессор, называемый каналом ввода-вывода (контроллер ввода-вывода), который осуществляет фактическое управление внешним запоминающим устройством при выполнении операций ввода и вывода данных. На центральный процессор теперь возлагается функция управления каналом ввода-вывода. Синхронизация параллельной работы центрального процессора и канала ввода/вывода осуществляется с применением системы прерываний. Канал через систему прерываний прерывает работу центрального процессора всякий раз при завершении операции ввода-вывода или при условии возникновения ошибок ввода/вывода.

Буферизация ввода-вывода основана на размещении между внешним и внутренним процессами одного или нескольких буферов, роль которых выполняют, как правило, непрерывные области первичной памяти.

Файл

Файловый способ хранения данных – это способ хранения данных, при котором каждый набор данных представляются как именованное, возможно, защищенное, собрание записей, называемой файлом.

Файл – идентифицированная совокупность экземпляров полностью описанного в конкретной программе типа данных, находящихся вне программы во внешней памяти и доступных программе посредством специальных операций.

Файловая система – система управления данными с файловым способом хранения.

Современные файловые системы ЭВМ предназначены для обслуживания многих тысяч файлов. Поэтому роль механизмов учета в составе файловой системы чрезвычайно важна. В настоящее время общепринятым приемом реализации механизма учета файлов является сведение всей учетной информации о расположении файлов в одну таблицу, называемую каталогом.

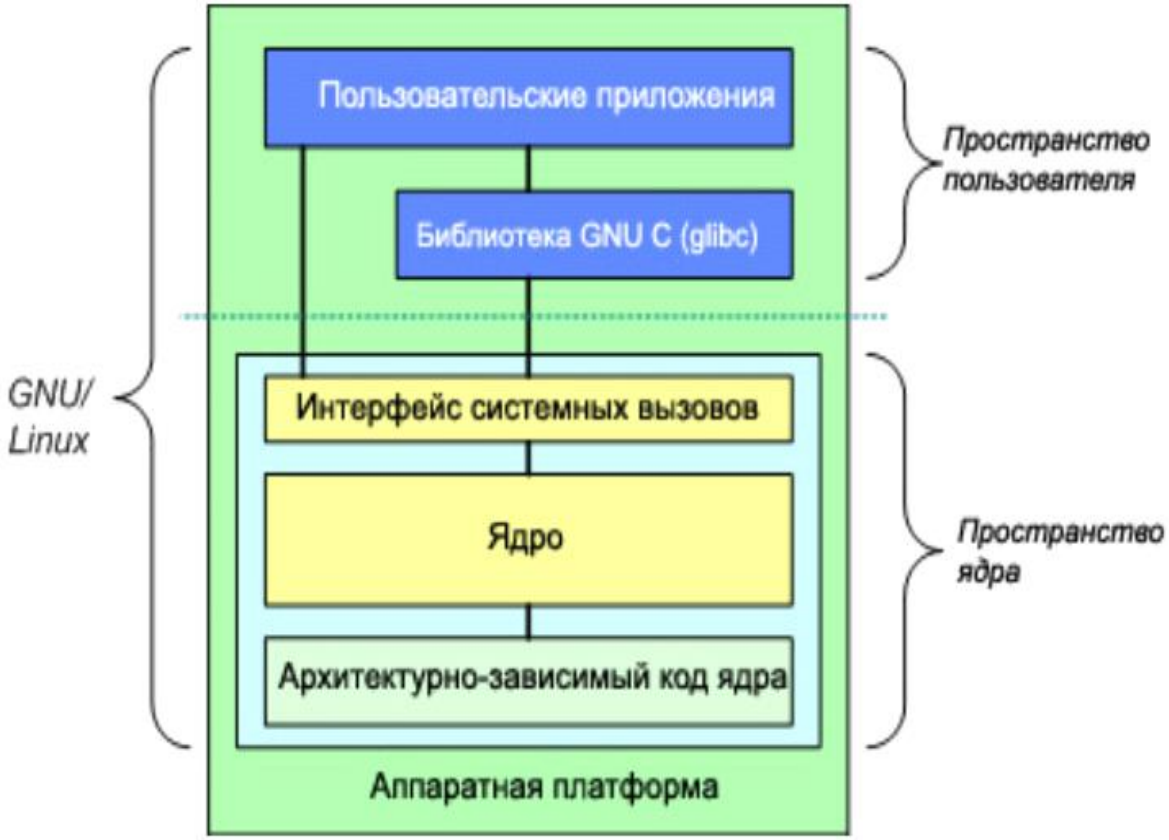
Система ввода-вывода

Драйвер устройства (Device driver) – программа, обеспечивающая взаимодействие операционной системы с физическим устройством.

Система ввода-вывода (Input-Output System) – часть операционной системы, обеспечивающая управление внешними устройствами, подключенными к ЭВМ.

Базовая система ввода-вывода (BIOS – Basic Input Output System) – часть программного обеспечения ЭВМ, поддерживающая управление адаптерами внешних устройств и представляющая стандартный интерфейс для обеспечения переносимости операционных систем между ЭВМ с одинаковым процессором. Базовая система ввода-вывода, как правило, разрабатывается изготовителем ЭВМ, хранится в постоянном запоминающем устройстве и рассматривается как часть ЭВМ.

Фундаментальная архитектура ОС GNU/Linux



Архитектура ядра Linux

SCI (System Call Interface) по сути представляет собой службу мультиплексирования/демультиплексирования вызова функций ядра из пространства пользователя.

Управление процессами сконцентрировано на исполнении процессов. В ядре эти процессы называются потоками (threads). Они соответствуют отдельным виртуализированным объектам процессора (код, данные, стек, процессорные регистры). В пространстве пользователя обычно

используется термин ***процесс.*** Ядро через SCI предоставляет механизмы для создания нового процесса (порождение копии, запуск на исполнение, вызов функции POSIX – Portable Operating System Interface), остановки процесса (kill, exit), взаимодействия и синхронизации между процессами (сигналы и механизмы POSIX).

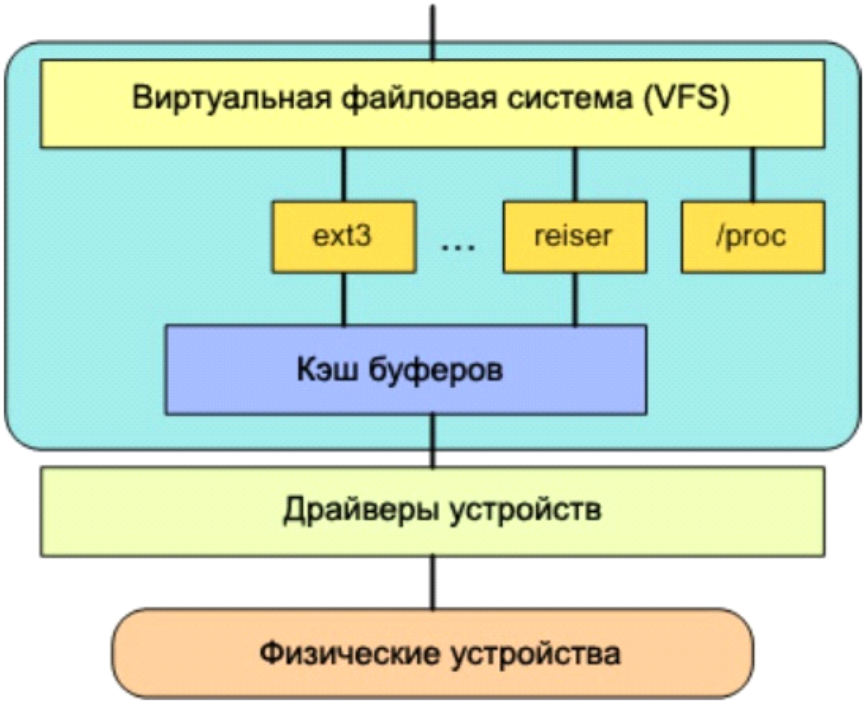
Еще одной задачей управления процессами является совместное использование процессора активными потоками. В ядре реализован алгоритм планировщика, время работы которого не зависит от числа потоков, претендующих на ресурсы процессора.

Управление памятью.

Память – еще один важный ресурс, которым управляет ядро. Память организуется в виде страниц (обычно по 4КБ). В условиях наличия большого числа потребителей памяти она может закончиться. Страницы при этом можно удалять из памяти и переносить на диск. Этот процесс называется подкачкой.

Виртуальная файловая система

VFS (Virtual File System) предоставляет общую абстракцию интерфейса к файловым системам. VFS обеспечивает коммутацию между SCI и файловыми системами, поддерживаемыми ядром.



На верхнем уровне располагается единая API-абстракция таких функций как открытие, чтение, закрытие, запись. На нижнем уровне

находятся абстракции файловых систем, которые определяют как реализуются функции верхнего уровня. Они представляют собой подключаемые модули для конкретных файловых систем (более 70).

Ниже уровня ФС находится кеш буферов, предоставляющий общий набор функций к уровню файловой системы. Ниже кеша буферов находятся драйверы устройств и конкретные физические устройства.

Сетевой стек

Имеет многоуровневую архитектуру, повторяющую структуру самих протоколов. В пространство пользователя через SCI предоставляется доступ к уровню сокетов. Уровень сокетов обеспечивает стандартный API к сетевой подсистеме.

Драйверы устройств

Подавляющее большинство кода ядра приходится на драйверы, которые позволяют работать с конкретными устройствами.

Архитектурно-зависимый код

Основная часть ядра Linux независима от архитектуры, на которой работает операционная система (Intel x86, Intel x64, ARM, MIPS, PowerPC, …).

Но в некоторых элементах для обеспечения работы и повышения производительности необходимо учитывать архитектуру.

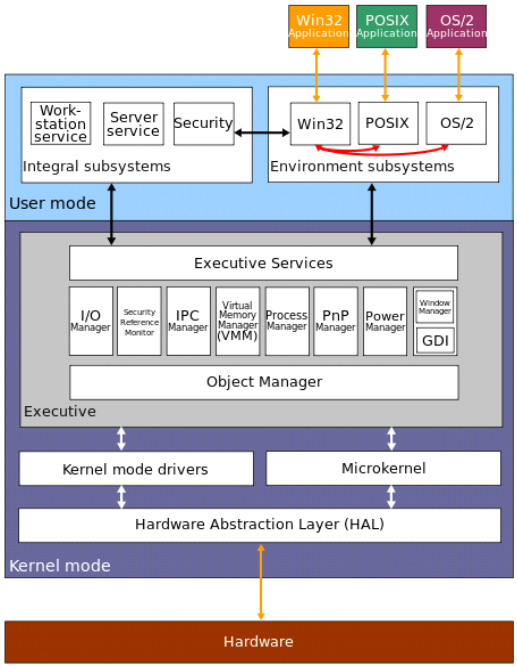
Гипервизор – операционная система для других операционных систем. В Linux встроен свой гипервизор Kernel-based Virtual Machine (KVM, виртуальная машина на базе ядра)

Порядок загрузки ОС Linux

Основные этапы загрузки:

- BIOS выполняет загрузочные шаги, специфичные для данного аппаратного обеспечения.
- Когда все устройства распознаны и правильно запущены, BIOS загружает и выполняет загрузочный код с одного из разделов заданного загрузочного устройства, который содержит фазу 1 загрузчика Linux. Фаза 1 загружает фазу 2 (значительный по размеру код загрузчика). Некоторые загрузчики могут использовать для этого промежуточный этап (под названием фаза 1,5), поскольку современные диски большого объёма могут некорректно считываться без дальнейшего кода.
- Загрузчик зачастую предлагает пользователю меню с доступными вариантами загрузки. Затем он загружает ядро, которое распаковывается в память, настраивает системные функции, такие как работа необходимого оборудования и управление страницами памяти, после чего делает вызов start_kernel().
- После этого start_kernel() выполняет основную настройку системы (прерывания, остальные функции управления памятью, инициализацию устройств, драйверов и т. д.), а потом порождает процесс бездействия, диспетчер и отдельно от них — процесс init (выполняющийся в пользовательском пространстве).
- Планировщик начинает более эффективно управлять системой, в то время как ядро переходит к бездействию.
- Процесс init выполняет необходимые сценарии, которые настраивают все службы и структуры, не относящиеся к уровню ядра, в результате чего будет создано пользовательское окружение, и пользователю будет предоставлен экран входа в систему.

Архитектура Windows NT



Архитектура WindowsNT имеет модульную структуру и состоит из двух основных уровней — компоненты, работающие в режиме пользователя, и компоненты режима ядра. Программы и подсистемы, работающие в режиме пользователя, имеют ограничения на доступ к системным ресурсам. Режим ядра имеет неограниченный доступ к системной памяти и внешним устройствам. Ядро системы NT называют гибридным ядром или макроядром.

Архитектура включает в себя само ядро, уровень аппаратных абстракций (HAL), драйверы и ряд служб (Executives), которые работают в режиме ядра (Kernel-mode drivers) или в пользовательском режиме (User-mode drivers).

Пользовательский режим Windows NT состоит из подсистем, передающих запросы ввода-вывода соответствующему драйверу режима ядра посредством менеджера ввода-вывода. Есть две подсистемы на уровне пользователя: подсистема окружения (запускает приложения, написанные для разных операционных систем) и интегрированная подсистема (управляет особыми системными функциями от имени подсистемы окружения). Режим ядра имеет полный доступ к аппаратной части и системным ресурсам компьютера. И также предотвращает доступ к критическим зонам системы со стороны пользовательских служб и приложений.

Режим ядра Windows NT имеет полный доступ к аппаратной части компьютера и системным ресурсам. Работает в защищенной области памяти. Контролирует потоки, управляет памятью и взаимодействием с аппаратной частью. Предотвращает доступ к критическим областям памяти со стороны приложений и служб пользовательского режима. Для выполнения подобных операций процесс пользовательского режима должен попросить режим ядра выполнить её от своего имени.

Архитектура x86 поддерживает 4 уровня привилегий — от 0 до 3, но используются только 0 и 3 уровень. Режим пользователя использует уровень 3, а режим ядра — 0. Это было сделано для возможности переноса на платформу RISC, которая использует только два уровня привилегий. Режим ядра состоит из исполнительных служб, которые представляют собой различные модули, выполняющие определенные задачи, драйвера ядра, само ядро и уровень аппаратных абстракций HAL.

Загрузка ОС Windows

BIOS – это набор микропрограмм, записанных в ПЗУ компьютера и служащих для инициализации устройств на материнской плате, их проверки и настройки, загрузки операционной системы.

При включении компьютера BIOS проверяет “железо” и если есть проблемы, то информирует нас звуковыми сигналами (набор длинных и коротких гудков). Вот таблица звуковых сигналов BIOS:

Сигналы AMI
Сигнал Возможная неисправность
Отсутствует Неисправен блок питания
2к Ошибка четности ОЗУ
3к Ошибка в первых 64 кБ ОЗУ
4к Неисправность системного таймера
5к Неисправен CPU
6к Неисправен контроллер клавиатуры
7к Неисправна системная плата
8к Неисправна память видеокарты
9к Ошибка контрольной суммы BIOS
10к Невозможна запись в CMOS
11к Неисправен кэш на системной плате
1д+2к Неисправна видеокарта
1д+3к Неисправна видеокарта
1д+8к Не подключен монитор

Сигналы AWARD
Сигнал Возможная неисправность
2к Обычно – проблемы в CMOS Setup или с системной платой (мелкие ошибки)
3д Ошибка контроллера клавиатуры
1д+1к Ошибки в ОЗУ
1д+2к Неисправна видеокарта
1д+3к Ошибка инициализации клавиатуры
1д+9к Ошибка при чтении из ПЗУ

к, повторяющийся Неисправен блок питания
д, повторяющийся Проблемы с ОЗУ
непрерывный Неисправен блок питания

После проверки BIOS считывает настройки из CMOS и в соответствии с мини стартует загрузчик с указанного носителя (CD, HDD, Flash карта). Если загрузка производится с жёсткого диска, то система считывает первые 512 байт Master Boot Record (MBR) и передаёт ему управление.

Если MBR не найден, то загрузка останавливается. Восстановить MBR можно с помощью консоли восстановления Windows (Recovery Console) командой fixmbr.

Загрузка старых версий Windows

Загрузкой Windows управляет NTLDR, который состоит из двух частей – первый StartUp переводит процессор в защищённый режим и стартует загрузчик ОС. Загрузчик содержит в себе основные функции для работы с дисками отформатированными в FAT*, NTFS и CDFS системы. Загрузчик считывает содержимое boot.ini и, в соответствии с его содержимым (количество ОС, диски на котором установленны и т.п.), продолжает загрузку. Если Windows была переведена в состояние гибернации, то NTLDR загружает в память компьютера файл hiberfil.sys и передаёт управление в ядро Windows. Если вы завершили работу компьютера простым выключением/перезагрузкой, то NTLDR загружает DOS’овский файл NTDETECT.COM, который строит список аппаратного обеспечения и загружает саму операционную систему Windows.

Если файл NTLDR удалён/перемещён/повреждён, то система не загрузится и выведет сообщение “NTLDR is missing. Press CTRL+ALT+DEL to restart“. Решить эту проблему можно в консоли восстановления Windows (Recovery Console) командой fixboot или скопировав NTLDR с рабочей системы в корень диска.

Перед загрузкой ядра, NTLDR выводит на экран опции запуска (Если была нажата клавиша F8, или работа системы была завершена аварийно). После выбора параметров запуска, стартует ядро системы – ntoskrnl.exe (мы видим анимацию из белых прямоугольников на чёрном экране).

Далее загружается тип абстрактного уровня аппаратного обеспечения – HALL.DLL. Это нужно, чтобы ядро могло абстрагироваться от железа, оба файла находятся в директории System32.

Далее загружается библиотека расширения ядра отладчика аппаратного обеспечения kdcom.dll и bootvid.dll, который загружает логотип Windows и индикатор статуса загрузки).

Одним из самых ответственных моментов является загрузка системного реестра config\system, очень часто система не может прочитать файл system и загрузка становится невозможна или начинается циклическая перезагрузка.

Процесс загрузки можно считать завершённым, если перед пользователем появилось окно входа в систему (инициализируется WINLOGON.EXE)

Процесс загрузки Windows Vista и Windows 7 (Seven) начинает отличатся от процесса загрузки предыдущих версий ОС уже после чтения MBR. Установщик Windows создаёт небольшой загрузочный раздел, в котором и находяться всё, что нужно для запуска ОС. MBR передаёт загрузку PBR (Partition Boot Record), а затем стартует BOOTMGR (Windows Boot Manager). BOOTMGR пришёл на смену NTLDR и руководит загрузкой операционной системы. BOOTMGR читает параметры загрузки из Boot Configuration Database (BCD, Базы данных загрузочной конфигурации, пришла на смену boot.ini) и загружает Winload.exe (OS loader boot application, загрузчик ОС). Winload.exe загружает ядро операционной системы, далее процесс загрузки похож на старт Windows XP.

Для редактирования Boot Configuration Database (BCD) можно использовать утилиту Bcdedit.exe запустив её из Windows Recovery Environment (WinRE). Там же можно задействовать утилиту Bootrec.exe для исправления ошибок.

Вывод

Для понимания работы механизмов защиты нужны фундаментальные знания по архитектуре ОС. В рамках данной лекции мы с вами повторили основные компоненты ОС. Рассмотрели архитектуру и порядок загрузки Windows и Linux.