**Лекция №17**

**Тема «Планирование и установка операционной системы»**

**Цель занятия:** Сформировать знания о разработке ОС, требованиях к операционным системам, о расширяемости, переносимости, совместимости, безопасности, семействах современных ОС.

### Цели проектирования и разработки ОС

Точки зрения пользователей и разработчиков ОС в данном отношении несколько различаются.

**Цели с точки зрения пользователя**: ОС должна быть удобной в использовании, простой для изучения, надежной, безопасной и быстрой.

**Цели с точки зрения разработчика ОС**: ОС должна быть несложной для проектирования, реализации и сопровождения, а также гибкой, надежной, свободной от ошибок и эффективной.

### Механизмы и политики

При разработке ОС, как и любой другой сложной программной системы, различают концепции **механизма**и **политики**.

**Механизм**определяет, **каким образом**реализовать некоторую функциональность (например, *виртуальную память*). **Политика**определяет, **что**именно требуется реализовать, т.е. набор требований, целей, условий для реализации функциональности. Например, **политика безопасности**в ОС определяет правила, по которым программе или пользователю предоставляются некоторые полномочия (например, возможность открытия некоторого файла). Эта политика может быть реализована многими способами, например, с помощью **списков управления доступом (access control lists** ); это понятие будет рассмотрено немного позже. Отделение механизма от политики – очень важный принцип для разработки ОС. Он допускает максимум гибкости, если "политические" решения могут быть изменены впоследствии.

### Реализация операционных систем

Традиционно, начиная с 1950-х гг., ОС разрабатывались на низкоуровневых языках – ассемблерах. Система UNIX была первой ОС, разработанной почти полностью на *языке высокого уровня* – Си. Впоследствии язык Си стал одним из наиболее популярных языков *системного программирования*. Благодаря новаторскому подходу авторов UNIX, сложилась тенденция разработки ОС на *языках высокого уровня*.

Преимущества использования *языков высокого уровня* очевидны: код на *языке высокого уровня*

1. может быть разработан быстрее
2. более компактен
3. легче для понимания и отладки.

Кроме того, операционная система гораздо легче **переносима** на другие аппаратные платформы, если она разработана на *языке высокого уровня*.

Гипотетический недостаток у *языков высокого уровня*, по сравнению с *ассемблерами*, только один – возможная неэффективность реализации высокоуровневых конструкций языка, по сравнению с "ручным" программированием на ассемблере. Однако данная проблема может быть решена путем разработки эффективного оптимизирующего компилятора с языка высокого уровня и не должна препятствовать его использованию.

Близкими автору примерами разработки ОС на *языках высокого уровня*, помимо UNIX, является разработка на Си ОС Solaris фирмой Sun, а также разработка ОС "Эльбрус" на *языке высокого уровня* Эль-76. Поставка ОС "Эльбрус" вместе с ее исходными кодами была весьма полезна для программистов и стимулировала целый ряд работ отраслевых организаций по разработке ОС реального времени на базе ОС "Эльбрус".

### Генерация операционной системы

ОС проектируются с целью использования на любом компьютере из некоторого класса. В поставку ОС входит очень большой набор типовых модулей для различных классов компьютеров (например, настольных и портативных). Поэтому для каждого компьютера система должна быть сконфигурирована при ее **инсталляции** – установке ОС на конкретный компьютер.

Программа генерации ОС получает информацию о специфической конфигурации компьютерной системы.

После генерации и инсталляции ОС система готова к работе.

Следующий этап - **загрузка (booting)**– запуск компьютера посредством загрузки ядра ОС. При запуске компьютера первой запускается **программа раскрутки (bootstrap program)**– код, хранящийся в постоянной памяти (ROM) базовой системы ввода-вывода (*BIOS*) компьютера. Код программы раскрутки находит ядро ОС, загружает его в память и запускает.

### Требования, предъявляемые к современным операционным системам

Операционная система создает среду для выполнения прикладных программ и во многом определяет, какими полезными для пользователя свойствами эти программы будут обладать, поэтому к операционным системам предъявляется ряд требований. Главными из них являются способность эффективно управлять ресурсами; обеспечивать удобный интерфейс для пользователя и прикладных программ; осуществлять мультипрограммную обработку; поддерживать виртуальную память, свопинг (операции выгрузки сегментов во внешнюю память и их повторная загрузка в основную память.), многооконный интерфейс и др. К современным

операционным системам предъявляются не менее важные рыночные требования: расширяемость, переносимость, надежность и отказоустойчивость, совместимость, производительность, безопас­ность (рис. 1).

|  |
| --- |
| http://edu.dvgups.ru/METDOC/ITS/STRPRO/ASY/METOD/UP/frame/1_5.files/image001.gif |

Рис. 1. Рыночные требования, предъявляемые к операционным системам

 1. ***Расширяемость***. Код операционной системы должен быть написан таким образом, чтобы при необходимости можно было легко внести дополнения и изменения, не нарушая целостности системы. В то время как аппаратная часть компьютера устаревает за несколько лет, полезная жизнь операционной системы может измеряться десятилетиями, например *ОС UNIX*. Операционные системы эволюционно модифицируются со временем, и эти трансформации более значимы, чем изменения аппаратных средств. Трансформации операционной системы представляют собой приобретение новых свойств, например, поддержки новых устройств, возможности связи с сетями нового типа, поддержки новых технологий. Причем сохранение целостности кода, какие бы изменения не вносились в операционную систему, является главной целью разработки.

Расширяемость может достигаться за счет *модульной структуры* операционной системы, при которой программы строятся из набора отдельных модулей, взаимодействующих через функциональный интерфейс. Новые компоненты добавляются в операционную систему модульным путем и выполняют свою работу, используя интерфейсы, поддерживаемые существующими компонентами. Использование объектов для представления системных ресурсов также улучшает расширяемость системы. *Объекты* – это абстрактные типы данных, над которыми можно производить только те действия, которые предусмотрены специальным набором объектных функций. Объекты позволяют единообразно управлять системными ресурсами. Добавление новых объектов не разрушает существующие объекты и не требует изменений существующего кода.

Прекрасные возможности для расширения предоставляет подход к структурированию операционных систем по типу клиент-сервер с использованием микроядерной технологии. В соответствии с этим подходом операционная система строится как совокупность привилегированной управляющей программы и набора непривилегированных услуг-сер­веров. Основная часть операционной системы может оставаться неизменной при добавлении новых серверов или улучшении старых. Другой возможностью расширить функциональные возможности операционной системы являются средства вызова удаленных процедур (*RPC*), которые могут добавляться в любую машину сети и немедленно поступать в распоряжение прикладных программ на других машинах сети.

Некоторые операционные системы для улучшения расширяемости поддерживают загружаемые драйверы, которые добавляются в систему во время ее работы. Новые файловые системы, устройства и сети могут поддерживаться путем написания драйвера устройства, драйвера файловой системы или транспортного драйвера и загрузки его в систему.

2. ***Переносимость***. Код операционной системы должен легко переноситься с процессора одного типа на процессор другого типа и с аппаратной платформы (которая включает наряду с типом процессора и способ организации всей аппаратуры компьютера) одного типа на аппаратную платформу другого типа. Требование переносимости кода тесно связано с расширяемостью. Расширяемость позволяет улучшать операционную систему, в то время как переносимость дает возможность перемещать всю систему на машину, базирующуюся на другом процессоре или аппаратной платформе, делая при этом по возможности небольшие изменения в коде. Операционные системы описываются либо как переносимые, либо как непереносимые. При написании переносимой операционной системы нужно следовать следующим правилам:

–   большая часть кода должна быть написана на языке, который имеется на всех машинах, обычно на языке высокого уровня, предпочтительно стандартизованном, например, на языке *С* (программа, написанная на Ассемблере, не является переносимой, если только ее не придется переносить на машину, обладающую командной совместимостью с той, для которой написана операционная система);

–   необходимо учитывать физическое окружение, в которое программа должна быть перенесена, так как различная аппаратура требует различных решений при создании операционной системы (например, операционная система, построенная на 32-битовых адресах, не может быть перенесена на машину с 16-битовыми адресами);

–   следует минимизировать или по возможности исключить части кода, которые непосредственно взаимодействуют с аппаратными средствами, так как зависимость от аппаратуры может иметь много форм (некоторые очевидные формы зависимости включают прямое манипулирование регистрами и другими аппаратными средствами);

–   если аппаратно зависимый код не может быть полностью исключен, то он должен быть изолирован в нескольких хорошо локализуемых модулях.

3. ***Надежность и отказоустойчивость***. Система должна быть защищена от внутренних и внешних ошибок, сбоев и отказов. Действия операционной системы должны быть предсказуемыми, и приложения не должны приводить к ее сбоям.

4. ***Совместимость***. Операционная система должна иметь средства для выполнения прикладных программ, написанных для других операционных систем, или для более ранних версий данной операционной системы, причем пользовательский интерфейс должен быть совместим с существующими системами и стандартами. Совместимость операционных систем рассматривается на двух уровнях:

–   *двоичной совместимости*, которая достигается совместимостью на уровне команд процессора, системных вызовов и на уровне библиотечных вызовов, если они являются динамически связываемыми;

–   *совместимости исходных текстов приложений*, что требует наличия соответствующего компилятора в составе программного обеспечения, а также совместимости на уровне библиотек и системных вызовов, при этом необходима перекомпиляция имеющихся исходных текстов в новый выполняемый модуль.

Совместимость на уровне исходных текстов важна для разработчиков приложений, для конечных пользователей практическое значение имеет только двоичная совместимость, благодаря которой один и тот же коммерческий продукт можно использовать в различных операционных средах и на различных машинах. Совместимость зависит от многих факторов, самый важный фактор – архитектура процессора. Если процессор, на который переносится операционная система, использует аналогичный набор команд (возможно с некоторыми добавлениями) и тот же диапазон адресов, тогда двоичная совместимость может быть достигнута достаточно просто. Совместимость между процессорами, основанными на разных архитектурах, требует написания специальных подпрограмм и использования прикладных сред. Учитывая, что основную часть программы, как правило, составляют вызовы библиотечных функций, прикладная среда имитирует библиотечные функции целиком, используя заранее написанную библиотеку функций аналогичного назначения, а остальные команды эмулирует каждую по отдельности.

Другим средством обеспечения совместимости программных и пользовательских интерфейсов является соответствие стандартам *POSIX*, которые правительственные агентства США начали разрабатывать во второй половине 80-х гг. ХХ в. в качестве стандартов на поставляемое оборудование при заключении правительственных контрактов в компьютерной области. *POSIX* – *собрание международных стандартов интерфейсов операционных систем в стиле* *UNIX*. Использование стандарта *POSIX* (*IEEE* стандарт 1003.1 - 1988) позволяет создавать програм­мы в стиле *UNIX*, которые могут легко переноситься из одной вычислительной системы в другую.

5. ***Безопасность***. Операционная система должна обладать средствами защиты ресурсов одних пользователей от других. В дополнение к стандарту *POSIX* правительство США определило требования компьютерной безопасности для приложений, используемых правительством, которые стали желаемыми свойствами для любой многопользовательской системы. Правила безопасности определяют такие свойства, как защита ресурсов одного пользователя от других и установление квот по ресурсам для предотвращения захвата одним пользователем всех системных ресурсов, например, памяти. Обеспечение защиты информации от несанкционированного доступа является обязательной функцией сетевых операционных систем. В большинстве популярных систем гарантируется степень безопасности данных, соответствующая уровню *С2*в системе стандартов США.

Основы стандартов в области безопасности были заложены правилами «Критерии оценки надежных компьютерных систем». Этот документ, изданный в США в 1983 г. национальным центром компьютерной безопасности (*NCSC* – *National Computer Security Center*), часто называют *Оранжевой книгой*. В соответствии с требованиями Оранжевой книги безопасной считается такая система, которая «посредством специальных механизмов защиты контролирует доступ к информации таким образом, что только имеющие соответствующие полномочия лица или процессы, выполняющиеся от их имени, могут получить доступ на чтение, запись, создание или удаление информации».

Иерархия уровней безопасности, приведенная в Оранжевой книге, помечает низший уровень *D*, высший – *А*. В класс *D*попадают системы, оценка которых выявила их несоответствие требованиям всех других классов. Основными свойствами уровня *С* являются: наличие подсистемы учета событий, связанных с безопасностью, и избирательный контроль доступа. Уровень *С* включает два подуровня: *С1* – обеспечивает защиту данных от ошибок пользователей; С2 – обеспечивает идентификацию пользователей путем ввода уникального имени и пароля перед доступом к системе.

Избирательный контроль доступа позволяет владельцу ресурса определить, кто имеет доступ к ресурсу и что он может с ним делать, путем предоставляемых прав доступа пользователю или группе пользователей. Средства учета и наблюдения (*auditing*) обеспечивают возможность обнаружить и зафиксировать важные события, связанные с безопасностью, или любые попытки создать, получить доступ или удалить системные ресурсы. Защита памяти обеспечивается инициализацией перед повторным использованием. На этом уровне система не защищена от ошибок пользователя, но поведение его может быть проконтролировано по записям в журнале, оставленным средствами наблюдения и аудитинга.

Системы уровня *В* основаны на помеченных данных и распределении пользователей по категориям, т. е. реализуют мандатный контроль доступа. Каждому пользователю присваивается рейтинг защиты, и он может получать доступ к данным только в соответствии с этим рейтингом. Этот уровень в отличие от уровня *С* защищает систему от ошибочного поведения пользователя. Уровень *А* является самым высоким уровнем безопасности, он требует в дополнение ко всем требованиям уровня *В* выполнения формального, математически обоснованного доказательства соответствия системы требованиям безопасности. Уровень безопасности *А* занимает своими управляющими механизмами до 90 % процессорного времени.

6. ***Производительность***. Система должна обладать настолько хорошим быстродействием и временем реакции, насколько это позволяет аппаратная платформа.

**Современные тенденции в развитии ОС**

На основе опыта использования многих современных ОС, можно выделить следующие основные тенденции в их развитии.

1. **Графические оболочки**. Любая современная ОС имеет графический пользовательский интерфейс, причем (по вполне понятным причинам острой конкуренции между фирмами-разработчиками) графические оболочки для всех ОС примерно одинаковы по возможностям. Подчас пользователю трудно сориентироваться, в какой именно ОС он работает, хотя для конечных пользователей (непрограммистов), по-видимому, такая унификация удобна.
2. **Поддержка новых сетевых технологий и Web-технологий**. Сети и Интернет активно развиваются. Появляются новые стандарты и протоколы – IPv6, HTML 5 (для облачных вычислений) и т.д. Современные ОС развиваются в направлении поддержки всех новых сетевых технологий.
3. **Усиленное внимание к механизмам безопасности и защиты**. Во многом благодаря инициативе Trustworthy Computing, начатой фирмой Microsoft в 2002 г., а также ввиду все усиливающейся киберпреступности, все современные ОС уделяют повышенное внимание безопасности: при просмотре веб-страниц браузеры выполняют их проверку на отсутствие phishing; загрузки и инсталляции программ из сети выполняются только с явного согласия пользователя и т.д.
4. **Поддержка многопоточности и многоядерных процессоров**. Ввиду широкого распространения многоядерных процессоров, все современные ОС имеют библиотеки программ, поддерживающие эту возможность аппаратуры. Именно благодаря многоядерной архитектуре, становится реально возможным параллельное выполнение потоков (threads).
5. **Поддержка распределенных и параллельных вычислений**. Современные ОС имеют в своем составе высокоуровневые библиотеки, позволяющие разрабатывать параллельные алгоритмы решения задач – например, поддерживающие стандарты параллелизма OpenMP и MPI.
6. **Виртуализация ресурсов и аппаратуры**. Современные ОС имеют в своем составе средства виртуализации, позволяющие выполнять приложения для других платформ в изолированных виртуальных машинах, в которые могут быть инсталлированы другие операционные системы.
7. **Развитие файловых систем** с целью защиты информации и значительного увеличения размера файлов (для мультимедиа). Современные требования обработки мультимедийной информации приводят к тому, что старые файловые системы (например, FAT) оказываются недостаточными для хранения мультимедийных файлов. Например, максимальный размер файла в системе FAT – 4 гигабайта – легко может быть превышен при переписи на компьютер цифровой видеопленки длительностью 10-15 минут. Поэтому разрабатываются новые файловые системы, допускающие хранение очень больших файлов, например, система ZFS в ОС Solaris. Другим требованием является обеспечение конфиденциальности информации, которое приводит к необходимости реализации в файловых системах возможности криптования (которая реализована, например, в файловой системе ZFS).
8. **Поддержка облачных вычислений** – совсем новая тенденция в развитии ОС, начало которой положила "облачная" ОС Windows Azure фирмы Microsoft.

### Контрольные вопросы:

1. *Охарактеризуйте цели проектирования и разработки ОС?*
2. *Что такое механизмы?*
3. *Как происходит генерация операционной системы?*
4. *Что такое расширяемость?*
5. *Что такое переносимость?*
6. *Что такое надежность и отказоустойчивость?*
7. *Что такое совместимость?*
8. *Что такое двоичная совместимость?*
9. *Что такое совместимости исходных текстов приложений?*
10. *Что такое безопасность?*
11. *Что такое производительность?*
12. *Перечислите современные тенденции в развитии ОС.*
13. *Охарактеризуйте каждое из направлений в развитии современных ОС.*