Операционные системы. Раздаточный материал.

1. Модель операционной системы, построенной на концепции абстрактной машины.

Данная модель имеет иерархическую структура и была впервые предложена Дейкстра.

Первой системой, построенной таким образом была простая пакетная система THE, которую построил Дейкстра и его студенты в 1968 году.

В иерархической структуре операционной системы отдельные функции выделяются согласно их сложности. Характерному масштабу времени выполнения, а также уровню абстракции. **Основная идея** заключается в том, чтобы создать такие уровни абстракции, чтобы на каждом уровне игнорировались подробности процессов происходящих на более низких уровнях.

Так, например когда интрепретатор команд загружает программу с диска в оперативную память – эта программа и соответственно интерпретатор может не указывать положение считывающих головок – это выполняется программами более низкого уровня.

№ уровня	Имя	Объекты	Примеры операций
1	электрические схемы	вентили, порты, триггеры, регистры	сбросить, очистить, передать, установить
2	система команд	стек для вычислений, массивы данных	загрузка, запись, переход, сложение, вычитание
3	программы	подпрограммные сегменты, стек вызовов	обращение к стеку вызовов, возврат из подпрограмм
4	прерывания	подпрограммы прерываний	активировать, маскировать, демаскировать
5	элементарные процессы	процессы, семафоры, списки готовности	ждать сигнал, приостановить, возобновить
6	локальная внешняя память	блоки данных, каналы устройств хранения	прочесть, записать, загрузить
7	виртуальная память	сегменты памяти	прочесть, записать, загрузить
8	конвееры /программные каналы	конвееры	создать уничтожить открыть записать закрыть
9	файловая система	файлы	создать уничтожить открыть записать закрыть, прочесть
10	устройства	внешние устройства, принтеры, плоттеры	создать уничтожить открыть записать закрыть, прочесть
11	каталоги	каталоги	связать, открыть, найти
12	процессы пользователя	процессы пользователя	выйти, уничтожить, приостановить возобновить

13	оболочка	программная среда пользователя	оператор на языке оболочки

Одним из важнейших принципов принятой данный гипотетической модели ОС является **независимость (для пользователя) операций ввода-вывода от внешнего устройства**.

.

2. Рыночные требования, предъявляемые к ОС

К этим требованиям относятся:

1. *Расширяемость*. Код должен быть написан таким образом, чтобы можно было легко внести дополнения и изменения, если это потребуется, и не нарушить целостность системы.
2. *Переносимость*. Код 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.
3. *Надежность и отказоустойчивость*. Система должна быть защищена как от внутренних, так и от внешних ошибок, сбоев и отказов. Ее действия должны быть всегда предсказуемыми, а приложения не должны быть в состоянии наносить вред ОС.
4. *Совместимость*. ОС должна иметь средства для выполнения прикладных программ, написанных для других операционных систем. Кроме того, пользовательский интерфейс должен быть совместим с существующими системами и стандартами.
5. *Безопасность*. ОС должна обладать средствами защиты ресурсов одних пользователей от других.
6. *Производительность*. Система должна обладать настолько хорошим быстродействием и временем реакции, насколько это позволяет аппаратная платформа.

2.1. Расширяемость

В то время, как аппаратная часть компьютера устаревает за несколько лет, полезная жизнь операционных систем может измеряться десятилетиями. Примером может служить ОС UNIX. Поэтому операционные системы всегда эволюционно изменяются со временем, и эти изменения более значимы, чем изменения аппаратных средств. Изменения ОС обычно представляют собой приобретение ею новых свойств. Например, поддержка новых устройств, таких как CD-ROM, возможность связи с сетями нового типа, поддержка многообещающих технологий, таких как графический интерфейс пользователя или объектно-ориентированное программное окружение, использование более чем одного процессора. Сохранение целостности кода, какие бы изменения не вносились в операционную систему, является главной целью разработки.

Некоторые ОС для улучшения расширяемости поддерживают загружаемые драйверы, которые могут быть добавлены 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.

2.2. Переносимость

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. Написание переносимой ОС аналогично написанию любого переносимого кода - нужно следовать некоторым правилам.

1. Во-первых, большая часть кода должна быть написана на языке, который имеется на всех машинах, куда вы хотите переносить систему.

2. Во-вторых, следует учесть, в какое физическое окружение программа должна быть перенесена.
3. В-третьих, важно минимизировать или, если возможно, исключить те части кода, которые непосредственно взаимодействуют с аппаратными средствами.
4. В-четвертых, если аппаратно зависимый код не может быть полностью исключен, то он должен быть изолирован в нескольких хорошо локализуемых модулях. Аппаратно-зависимый код не должен быть распределен по всей системе.

2.3. Совместимость

Одним из аспектов совместимости является способность ОС выполнять программы, написанные для других ОС или для более ранних версий данной операционной системы, а также для другой аппаратной платформы.
Необходимо разделять вопросы двоичной совместимости и совместимости на уровне исходных текстов приложений.

- Двоичная совместимость достигается в том случае, когда можно взять исполняемую программу и запустить ее на выполнение на другой ОС. Для этого необходимы: совместимость на уровне команд процессора, совместимость на уровне системных вызовов и даже на уровне библиотечных вызовов, если они являются динамически связываемыми.
- Совместимость на уровне исходных текстов 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.

Соответствие стандартам POSIX также является средством обеспечения совместимости программных и пользовательских интерфейсов. Во второй половине 80-х правительственные агентства США начали разрабатывать POSIX("интерфейс переносимой ОС, базирующейся на UNIX" пер с англ) как стандарты на поставляемое оборудование при заключении правительственных контрактов в компьютерной области. Использование стандарта POSIX (IEEE стандарт 1003.1 - 1988) позволяет создавать программы стиле UNIX, которые могут легко переноситься из одной системы в другую.

2.4. Безопасность

В дополнение к стандарту POSIX правительство США также определило требования компьютерной безопасности для приложений, используемых правительством. Многие из этих требований являются 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 (таких как память).
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 "*Критериями оценки надежных компьютерных систем*". Этот документ, изданный в США в 1983 году национальным центром компьютерной безопасности (NCSC - National Computer Security Center), часто называют Оранжевой Книгой.
Иерархия уровней безопасности, приведенная в Оранжевой Книге, помечает низший уровень безопасности как D, а высший - как А.
А-уровень безопасности занимает своими управляющими механизмами до 90% процессорного времени. Более безопасные системы не только снижают эффективность, но и существенно ограничивают число доступных прикладных пакетов, которые соответствующим образом могут выполняться в подобной системе. Например для ОС Solaris (версия UNIX) есть несколько тысяч приложений, а для ее аналога В-уровня - только сотня.

2.5. Параметры надежности компьютерной техники

Reliability Надежность - Способность системы выполнять заданную функцию при заданных условиях в заданном объеме в течение данного периода времени.
Мера: **MTBF** (Mean Time Between Failure) - наработка на отказ, среднее время наработки на отказ, среднее время между отказами # средний интервал времени между отказами ремонтопригодного продукта. Характеристика надёжности оборудования и компонент, определяемая его производителями, как правило, в часах. Чем больше это время, тем лучше
Availability Готовность - Вероятность того, что система или оборудование будут удовлетворительно и эффективно функционировать в произвольный момент времени.
Мера**: Коэффициент готовности K**=MTBF/(MTBF+MTTR), где MTTR (Mean Time to Repair) - среднее время восстановления

3. Основные понятия

3.1.Системные вызовы

В любой операционной системе поддерживается некоторый механизм, который позволяет пользовательским программам обращаться за услугами ядра ОС. В ОС UNIX такие средства называются системными вызовами.
Системные вызовы (system calls) – это интерфейс между операционной системой и пользовательской программой. Они создают, удаляют и используют различные объекты, главные из которых процессы и файлы. Пользовательская программа запрашивает сервис у операционной системы, осуществляя системный вызов. С точки зрения программирования системный вызов похож на обычный вызов подпрограммы.

3.2. Прерывания

Прерывание (hardware interrupt) событие, генерируемое внешним (по отношению к процессору) устройством. Посредством аппаратных прерываний аппаратура либо информирует центральный процессор о том, что возникло какое-либо событие, требующее немедленной реакции (например, пользователь нажал клавишу), либо сообщает о завершении асинхронной операции ввода-вывода (например, закончено чтение данных с диска в основную память).

3.3. Исключительные ситуации

Исключительная ситуация (exception) событие, возникающее в результате попытки выполнения программой недопустимой команды, доступа к ресурсу при отсутствии достаточных привилегий или обращения к отсутствующей странице памяти. Исключительные ситуации так же, как и системные вызовы, являются синхронными событиями, возникающими в контексте текущей задачи. Исключительные ситуации можно разделить на исправимые и неисправимые.

3.4. Файлы

Файлы предназначены для хранения информации на внешних носителях, то есть, принято, что информация, лежащая, например, на диске, должна находиться внутри файла. Обычно под файлом понимают часть пространства на носителе информации, имеющую имя.
Главная задача файловой системы (file system) скрыть особенности ввода-вывода и дать программисту простую абстрактную модель файлов, независимых от устройств. Для чтения, создания, удаления, записи, открытия и закрытия файлов также имеется обширная категория системных вызовов (create, delete, open, close, read, write)..