04.09.2020

**История развития операционных систем**

* Первый этап 1945-1955 год. Ламповые машины. Операционные системы отсутствовали.
* Второй этап 1955-начало 60х. Полупроводниковые элементы – транзисторы. Происходит разделение персонала на программистов, операторов и специалистов по эксплуатации. Формируются задания на перфокартах, которые передаются вычислительной машине на выполнение. Для повышения производительности вычислительной машины формируется пакет заданий, которыми должна управлять управляющая программа. Появляются современные системы пакетной обработки, которые автоматизируют запуск одной программы из пакета за другой. Система пакетной обработки – прообраз современных ОС.
* Третий этап начало 60х-1980 год. Интегральные микросхемы. На смену перфокартам приходят магнитные ленты – устройства последовательного доступа, информация считывается в том порядке, в котором была записана. На смену магнитным лентам приходят магнитные диски – устройства прямого доступа, информацию можно считывать в любом порядке, это приводит к тому, что можно планировать задания. Появляется мультипрограммирование – одна программа выполняется операцию ввода-вывода и процессор не простаивает.

Для реализации мультипрограммирования должна быть соответствующая аппаратная поддержка:

* Защитные механизмы памяти
* Наличие привилегированных и непривилегированных команд
* Наличие привилегированного пользовательского режима поддержки
* Наличие прерывания

Для реализации мультипрограммирования в операционную системы должны быть заложены механизмы:

* Системные вызовы
* Планирование заданий
* Сохранение контекста прерванной задачи
* Синхронизация и коммуникация процессов
* Системные вызовы

*Система разделения времени* – процессор переключается между задачами по истечению определённого интервала времени.

1969 год – разработка UNIX.

* Четвертый этап. 1980 – наше время. Большие интегральные схемы (БИС). Появляются персональные компьютеры и классические сетевые распределенные операционные системы.

В 1983 году протокол TCP/IPстал стандартом.

07.09.2020

**Назначение и функции операционных систем**

Назначение ОС – организация вычислительного процесса

Операционная система – это совокупность программных средств, которые обеспечивают управление аппаратной частью компьютера, прикладными программами, а также взаимодействие их между собой и пользователем.

Задачи операционной системы:

* Увеличение пропускной способности (многозадачность)
* Уменьшение времени реакции системы на запросы пользователей
* Упрощение работы разработчиков, обслуживающего персонала (ПО, языки программирования, различные сервисные программы)

Функции ОС:

* Предоставление пользователю или программисту вместо реальной аппаратуры компьютера – расширенные виртуальные машины (высокоуровневые функции)
* Управление ресурсами (неразделяемые ресурсы, разделяемые ресурсы). Ресурс – любой физический и логический компонент системы. (логические компоненты – сетевые подключения, таблица выполняемых процессов; физические – процессор, оп).

Неразделяемые ресурсы - могут быть использованы на заданном отрезке времени только одним процессом. (файл, открытый на запись).

Разделяемые ресурсы – использование этого ресурса несколькими процессами.

Виды управлений:

* Управление процессами – создание, уничтожение процессов.
* Управление памятью – защищает адресное пространство процессов, выделяет память процессам, организует виртуальную память.
* Управление файлами (файловая система размещает данные на диске, что позволяет организовать работу диска)
* Управление внешними устройствами через драйверы
* Защита данных администрирования
* 1 рубеж защиты – login/password
* 2 рубеж защиты – аудит ОС (все события, происходящие в системе)
* Резервное копирование данных
* Пользовательский интерфейс (взаимодействие пользователя -ПК)

Требования к современным ОС

* Модульность
* Расширяемость – внесение изменений и дополнений без нарушения целостности системы
* Переносимость – код операционной системы должен легко переноситься с одного типа процессора на другой
* Совместимость – должна иметь средства для выполнения прикладных программ, написанных ля других операционных систем.
* Надежность и отказоустойчивость – должна быть защищена от внутренних и от внешних ошибок
* Высокая производительность

Классификация ОС

* 1 класс – по числу одновременно выполняемых задач. Однозадачные системы – в каждый момент времени может существовать не более, чем один пользовательский процесс, но одновременно с этим могут работать системные процессы. Многозадачные (MS DOS, Windows, Linux) – обеспечивают одновременное выполнение нескольких пользовательских процессов. Эта система решает проблему распределения ресурсов и конкуренции за эти ресурсы. В многозадачных системах отсутствуют защитные механизмы и средства коммуникации.
* 2 класс –по числу одновременно работающих пользователей.
* Однопользовательские – полный доступ ко всем системам компьютера.
* Многопользовательские – несколько пользователей могут одновременно использовать ресурсы компьютера. Многопользовательские системы являются многозадачными, но не наоборот.
* 3 класс – по типу доступа к компьютеру.

Системы пакетной обработки – формируется пакет, который предоставляется вычислительной машине, пользователь не взаимодействует с ПК.

Система разделения времени – переключение между задачи осуществляется по истечению определенного интервала времени.

Системы реального времени – должны обеспечить гарантированное время ответа на внешние события. Бывают мягкие и жесткие.

* Жесткие – недопустимость никаких задержек при никаких условиях. Бесполезность результатов при опозданиях. Катастрофа при задержке реакции. Цена опоздания бесконечно велика. QNX- ОС реального времени.
* Мягкие системы – за опоздание результатов приходится платить, но цена не слишком велика.
* 4 класс – по типу средств вычислительной техники для управления ресурсами которых предназначена ОС.
* Однопроцессорные ОС – все задачи выполняются на одном ПК
* Многопроцессорные ОС – команды выполняются на разных ПК. Параллельное выполнение команд.

Многопроцессорные ОС разделяются на ассиметричные и симметричные.

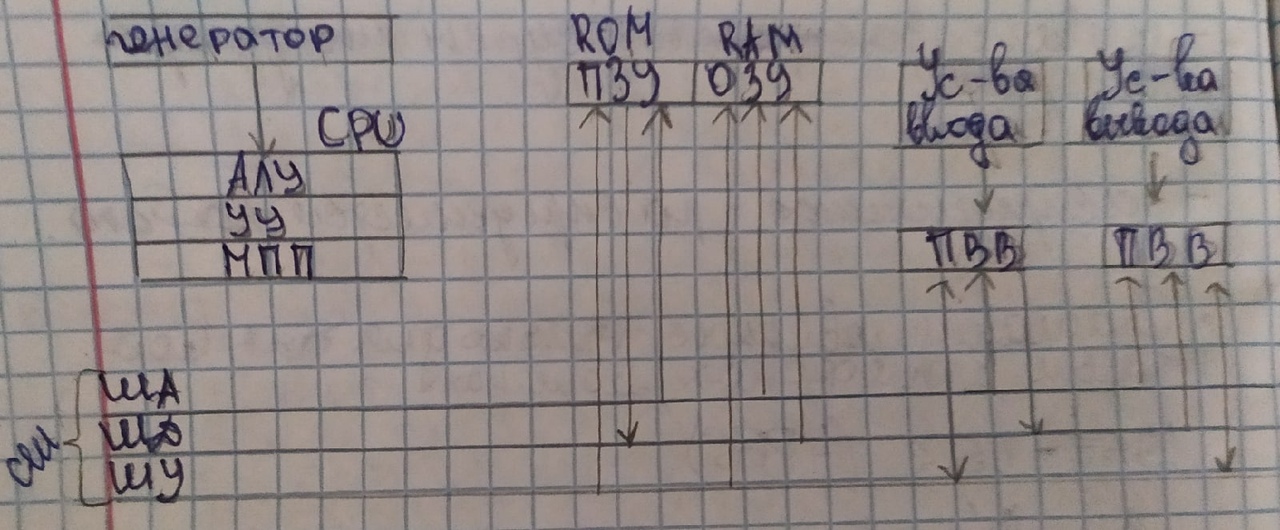
* Симметричные – на каждом процессе функционирует одно и тоже ядро, и задача может быть выполнена на любом процессоре.
* Ассиметричные – процессоры неравноправны. Существует главный процессор и подчиненные.
* Сетевые ОС – распределяют ресурсы
* Распределенные – распределяется работа по различным ПК
* 5 класс – специализированные. Бытовая техника, телефоны, машины.

ВидыОС: Dos, OS/2, Linux, Windows.

Архитектура персонального компьютера

Современные компьютеры построены на основе разработанных принципов Джона Фон Неймана:

* Принцип программного управления – программа состоит из набора команд, которые выполняется процессором автоматически друг за другом в определенной последовательности;
* Принцип однородности памяти – программы и данные хранятся в одних и тех же ячейках памяти;
* Принцип адресности – основная память состоит из пронумерованных ячеек и процессору в производный момент времени доступна любая ячейка.



Микропроцессор характеризуется быстродействием (кол-во операций, выполняемое процессором в секунду), разрядность (макс кол-во разрядов двоичного кода, которые могут передаваться и обрабатываться одновременно), архитектура – система команд, которые может выполнять микропроцессор, и кол-во ядер.

Микропроцессор состоит из:

* АЛУ - арифметико-логическое устройство, которое выполняет все арифметические и логические операции над числовой и символьной информацией;
* УУ – устройство управления, формирует и подает во все блоки компьютера в нужные моменты времени управляющие импульсы, которые поступают с генератора тактовой частоты.
* МПП – микропроцессорная память, служит на кратковременные хранения записи и выдачи информации, непосредственно используемой в вычислениях в ближайшее время работы компьютера. Строится на регистрах – быстродействующие ячейки памяти.
* Системная шина – обеспечивают сопряжение и связь всех устройств между собой. Обеспечивает 3 направления:
* Между микропроцессором (CPU)и основной памятью
* Между микропроцессором (CPU)и портами ввода-выводы
* Между основной памятью и портами ввода-вывода внешних устройств
* Включает в себя 3 шины:
* Шина адреса – выбор устройства или ячейки памяти, куда перемещаются или откуда считываются данные
* Шина данных – перемещаются данные между различными устройствами
* Шина управления – по ней передаются сигналы (чтения или записи), определяющие характер обмена информации.

ПЗУ (ROM) – предназначено для хранения информации и выдачи ее. Энергонезависимое. В ПЗУ хранятся: тестирующие программы, универсальный загрузчик.

ОЗУ (RAM) – энергозависимое. Предназначено для записи, считывания и хранения выполняемых программ и данных.

КЭШ – используются при обмене данными между процессором и оперативной памятью для компенсации разницы скорости обработки информации процессором и менее быстродействующей оперативной памятью.

VRAM– хранится изображение в закодированном виде

ВЗУ – служит для долговременного хранения данных

Порты ввода-вывода – служат для обеспечения обмена информацией компьютера с внешними устройствами

COM порт – последовательный порт. Передаются электрические импульсы, несущие информацию в машинном коде последовательно, один за другим.

LPT Порт – параллельный порт. Одновременная передача 8 электрических импульсов, несущих информацию в машинном коде.

USB порт – универсальная последовательная шина.

08.09.20

Принцип построения операционных систем.

Структура операционных систем.

Модуль ОС разделяются на 2 группы:

* Ядро - модули, выполняющие основные функции ОС
* Модули, выполняющие вспомогательные функции ОС
* Модули ядра ОС выполняют следующие базовые функции ОС:
* Управление процессами
* Управление памятью
* Управление устройствами ввода-вывода
* Ядро обеспечивает решение задачи организации *вычислительного процесса*: переключение контекстов, загрузка/выгрузка страниц, обработка прерываний и тд.
* Другая задача - поддержка приложений, создание для них *прикладной программной среды*. Приложения обращаются к ядру с запросами (*системными вызовами*) для выполнения базовых операций (открытие и чтение файла, вывод информации на дисплей и дт)
* Функции выполняемы ядром ОС требуют высокой скорости выполнения и для этого размещаются постоянно в оперативной памяти (*резидентные модули*)
* Вспомогательные модули выполняют полезные, но менее обязательные функции. Например:
* Архивирование информации
* Дефрагментация данных на диске.
* Поиск необходимого файла.

Вспомогательные модули Ос являются транзитными (располагаются в оперативной памяти только на время выполнения своих функций), оформлены как обычные приложения.

Вспомогательные модули ОС:

* Утилиты – приложения, решающие отельные задачи управления и сопровождения ОС
* Системные обрабатывающие программы – текстовые и графические редакторы, компиляторы, компоновщики и тп
* Программы предоставления пользователю дополнительных услуг – специальный вариант пользовательского интерфейса, калькулятор и тд.
* Библиотека процедур – модули различного назначения, упрощающие разработку приложений.

Вспомогательные модули обращаются к функциям ядра Ос через системные вызовы.

Процессор поддерживает 2 режима работы: пользовательский (user mode), привилегированный ( kernel mode)

Яро работает в привилегированном режиме.

Вспомогательные модули и приложения, запущенные пользователем, работают в пользовательском режиме.

Типы ядер ОС:

* Монолитное ядро – все компоненты ОС работают в одном адресном пространстве. Все компоненты ядра являются составными частями одной программы, используют общие структуры данных и взаимодействуют друг с другом путём непосредственного вызова процедур. Ядро совпадает со всей системой. Пример: UNIX
* Достоинства - скорость работы
* Недостатки – сбой в одном компоненты ядра приводит к нарушению работоспособности системы. При добавлении новых компонентов системы требуется перекомпиляция ядра.
* Модульное ядро – это современная усовершенствованная модификация архитектуры монолитных ядер ОС.
* Отличие: не требуется перекомпиляция при изменении состава.
* Микроядро – ядро минимизировано, выполняет только основные функции управления процессами и минимальный набор для работ с оборудованием. Большинство компонентов ОС оформлены в виде приложений, которые называются сервисами и работают в пользовательском режиме. Это клиент серверная архитектура.
* Достоинства: устойчивость к сбоям оборудования, сбоям системы; высокая степень модульности ядра; загружать и выгружать новые драйверы, не перезапуская системы; надёжность
* Недостатки: передача данных между процессорами требует накладных расходов.
* Экзо ядро – модифицированная структура микроядра. Минимизирован набор основных компонентов ядра, всё выполняется как пользовательские процессы.
* Нано ядро – выполняет лишь обработку аппаратных прерываний. Пример: keykos
* Гибридное ядро – сочетает особенности монолитных и микро-ядер. Основные компоненты помещаются в ядро, остальные компоненты оформлены в виде приложений и выполняются в пользовательском режиме.

Архитектура ОС:

* ОС на основе монолитного ядра.
* Многослойная (многоуровневая) структура – нижний слой (аппаратура), средний слой (ядро), верхний слой(утилиты, приложения). Пользовательский режим. Вся вычислительная система разбивается на слои, и ниже лежащий слой является основой для выше лежащего.
* Достоинства: хорошо тестируются и модифицируются
* Недостатки: сложно определить функции слоёв

Структура ядра:

1 слой - средства аппаратной поддержки, устройства которые участвуют в вычислительном процессе

2 слой – машинно-зависимые (отражают специфику аппаратной платформы, этот слой экранизирует верхние слои от особенностей аппаратуры, обеспечивает переносимость с одной платформы на другую)

З слой – базовые механизмы ядра( они не принимают решения переключений ресурсов, выполняют примитивные операции, только обрабатывают принятые решения)

4 слой – менеджер ресурсов – управление основными ресурсами вычислительной системы

5 слой – интерфейс системных вызовов – взаимодействие с приложениями, системными утилитами, образуя интерфейс прикладного программирования.

* Микро ядерная архитектура ОС

Суть: в привилегированном режиме работает небольшая часть операционной системы – микроядро. Оно защищено от остальных частей ОС и приложений. Состав: машинно-зависимые модули, а так же модули, выполняющие базовые функции ядра по управлению процессами, обработке прерываний, управление виртуальной памятью, пересылки сообщений, управление устройствами ввода-вывода. Все остальные более высокоуровневые функции ядра оформляются в виде приложений, работающих в пользовательском режиме. Взаимодействие между компонентами и приложения ОС осуществляется через передачу сообщений через микроядро. Компоненты операционной системы, выполняющиеся в пользовательском режиме, называются серверами ОС. Их задача обслуживание запросов локальных приложений и других модулей ОС.

Клиент, которым может быть либо прикладная программа, либо другой компонент ОС запрашивает выполнение некоторой функции у соответствующего сервера, посылая ему сообщение.

* Достоинства: переносимость; расширяемость; надёжность; поддержка распределённых вычислений
* Недостаток: низкая производительность; выполнение системного вызова сопровождается 4-мя переключениями (вместо 2х, как в классическом).
* Смешанная ОС

Большинство современных ОС используют различные комбинации этих подходов (монолитная, микро ядерная). Linux – монолитная система, с элементами микроядерной архитектуры.

Виды прерываний, обработка прерываний

Прерывание – это ситуация, приводящая к временному или окончательно прекращению выполнения последовательности команд одной и переходу к выполнению команд другой программы / или реакция на какое-либо событие

Классификация прерываний:

* Внешние (аппаратные) *-* могут возникать в результате поступления сигналов от аппаратных устройств, являются асинхронными. Обрабатывается соответствующими внешними устройствами.
* Внутренние (исключения) – происходят синхронно выполнению программы при появлении аварийной ситуации в ходе исполнения некоторой инструкции программы. Пример: деление на ноль, обращение к несуществующему адресу. Процессор при переходе на выполнение исключительной ситуации сохраняет часть своего состояния перед выполнением *текущей* команды. Исключительные ситуации возникают *синхронно* с работой процессора, но *непредсказуемо* для программиста, если только тот специально не заставил процессор делить некоторое число на ноль. Прерывания бывают исправимые и неисправимые. Исправимые – относятся такие исключительные ситуации, как отсутствие нужной информации в оперативной памяти. Неисправные – исключительные ситуации обычно возникают в результате ошибок в программах. Обычно операционная система реагирует на такие ситуации завершением программы, вызвавшей исключительную ситуацию. Обрабатываются специальными модулями ядра.
* Программные (системные вызовы) – когда одна программа требует сервис со стороны другой программы. Взаимодействие осуществляется через системные вызовы. Это удобный способ вызова процедур ОС. Обрабатываются обслуживающими системными вызовами.

Способы выполнения прерываний:

* *Векторный* – в процессоре передаётся номер вызываемой процедуры обработки прерывания.
* *Опрашиваемый* – процессор вынужден последовательно опрашивать потенциальные источники запроса прерывания.

Каждому возможному прерыванию процессора соответствует некоторый фиксированный адрес физической оперативной памяти. Каждое прерывание имеет свой номер (IRO) и с ним связана определенная подпрограмма.

Вектор прерываний – это адрес ячейки памяти, где находится программа по обработке данного прерывания.

Если произошло прерывание, то:

* Управление передается ОС
* ОС запоминает состояние прерванного процесса(контекст)
* ОС анализирует тип прерывания и передает управление соответствующей программе обработки этого прерывания
* После обработки прерывания процессор возвращается к выполнению прерванного процесса либо начинает обслуживать процесс с наивысшим приоритетом.

10.0.20

Структуры сетевой или распределенной ОС

Сетевые операционные системы:

*Сетевая операционная система* – комплекс программных модулей, предназначенных для повышения эффективности аппаратных ресурсов компьютера, путем рационального управления его ресурсами и разделения ресурсов между множеством выполняемых процессов. ( ОС со встроенными возможностями для работы в сети. (Программа, дополненная сетевыми возможностями.)

Сетевые возможности:

* Поддержка сетевого оборудования,
* поддержка сетевых протоколов,
* поддержка протоколов маршрутизации,
* поддержка фильтрации сетевого трафика,
* поддержка доступа к удалённым доступам по сети,
* поддержка сетевых протоколов авторизации,
* наличие в системе сетевых служб

Компоненты сетевой ОС:

* Средства управления локальными ресурсами компьютера (управление процессами, оперативной памятью)
* Сетевые средства, разделяемы на три компонента:
* Серверная часть ОС – средства предоставления локальных ресурсов и сервисов в общее пользование
* Клиентская часть ОС – средства запроса на доступ к удаленным ресурсам компьютера
* Транспортные средства ОС – совместно со 2-й системой обеспечивает передачу сообщений между компьютерами

Сетевые службы и сервисы:

* *Сетевой службой* называется совокупность серверной и клиенткой частей ОС, предоставляющих доступ к конкретному типу ресурса компьютера через сеть
* *Сервис* – интерфейс между потребителем услуг (пользователем или приложением) и поставщиком услуг (службой)

Подходы к построению сетевых операционных систем:

* Сетевые службы глубоко интегрированы в операционную систему (например, Windows NT)
* Сетевые службы объединены в виде некоторого набора программных модулей – оболочки ( например,LAN Server, NetWare for UNIX)
* Сетевые службы разрабатываются и поставляются в виде отдельных программных модулей (NDS для различных ОС)

Типы сетевых ОС:

* В зависимости о распределения функций между компьютерами, они могут выступать в роли выделенного сервера и клиентского узла

Сеть может быть построена по следующим схемам:

* На основе компьютеров, совмещающих функции клиента и сервера – *одноранговая связь*
* На основе клиентов и серверов – *сеть с выделенными серверами*
* Сеть, включающая узлы разных типов – *гибридная сеть*

В сетях с выделенными серверами используются серверные ОС для серверов.

Отличие серверных ОС от одноранговых ОС:

* Поддержка мощных аппаратных платформ
* Поддержка большого числа одновременно выполняемых процессов
* Мощные средства для администрирования сети
* Широкий набор сетевых служб

Для передачи сообщений по сети необходим протокол. TSP\IPE (стек протоколов)

Разрешения имён:

Система доменных имён (DNS) – является стандартным методом отображения IP-адресов на имена.

**Понятие интерфейса. Типы интерфейса.**

Интерфейс – система связей и взаимодействия устройств компьютера; средства взаимодействия пользователей с ОС компьютера.

Интерфейс пользователя – элементы и компоненты программы, которые способны оказать влияние на взаимодействие пользователей с программным обеспечением:

* Средство отображения информации
* Командные режимы
* Устройство и технологии ввода данных
* Диалоги
* Взаимодействия между пользователем и компьютером
* Обратная связь с пользователем

Виды пользовательского интерфейса:

* В период пакетной обработки заданий, интерфейс реализовывался с применением специальных носителей информации – перфокарт. Интерфейс командой строки (CLI)
* Взаимодействие пользователя с компьютером осуществляется путём ввода с клавиатуры команд
* В современных операционных системах командный режим обеспечивается с помощью эмуляторами. (cmd.exe(32), Command.com (16), exterm (linux))
* Для операционных систем с алфавитно-цифровым интерфейсом разрабатывались специальные оболочки (NC.EXE , VC.COM)

Графический интерфейс  
Интеллектуальный интерфейс – приближен к человеческой форме общения.

Программный интерфейс – интерфейс прикладного программирования

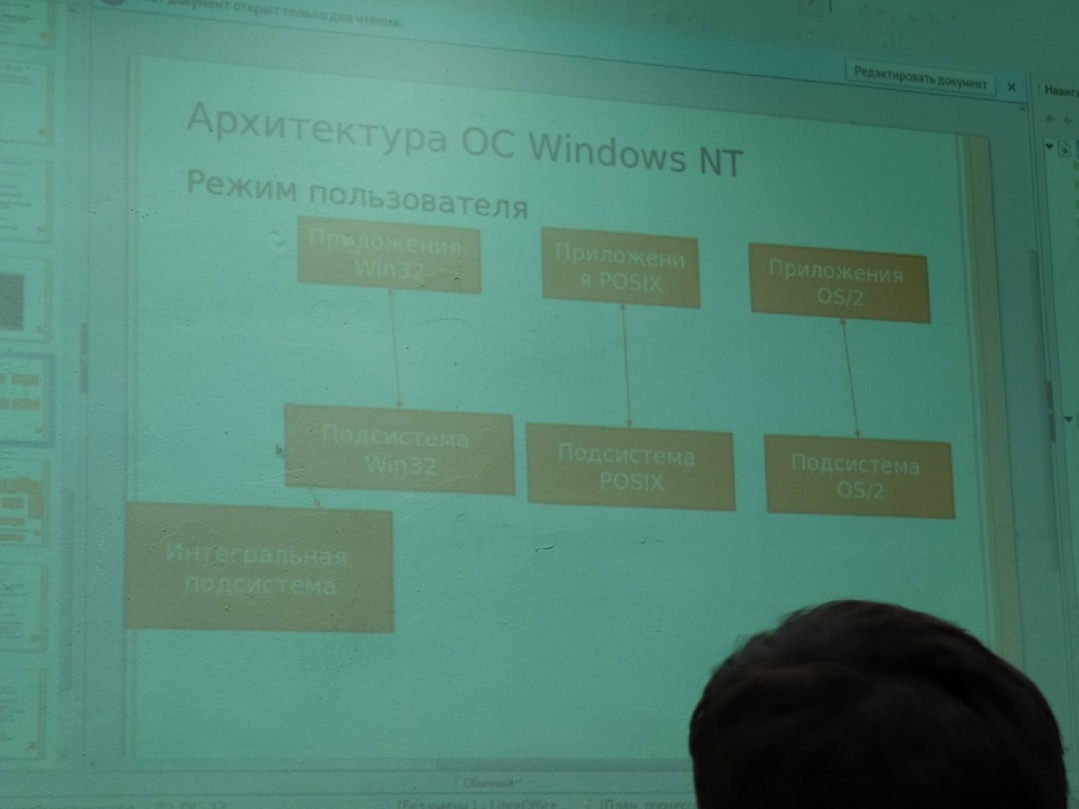
При написании приложений программист пользуется набором сервисных функции ОС, которые называются интерфейсом прикладного программирования. От пользователя эти функции скрыты за оболочкой алфавитно-цифрового или графического пользовательского интерфейса. Приложение выполняет обращение к функциям IP с помощью системных вызовов.

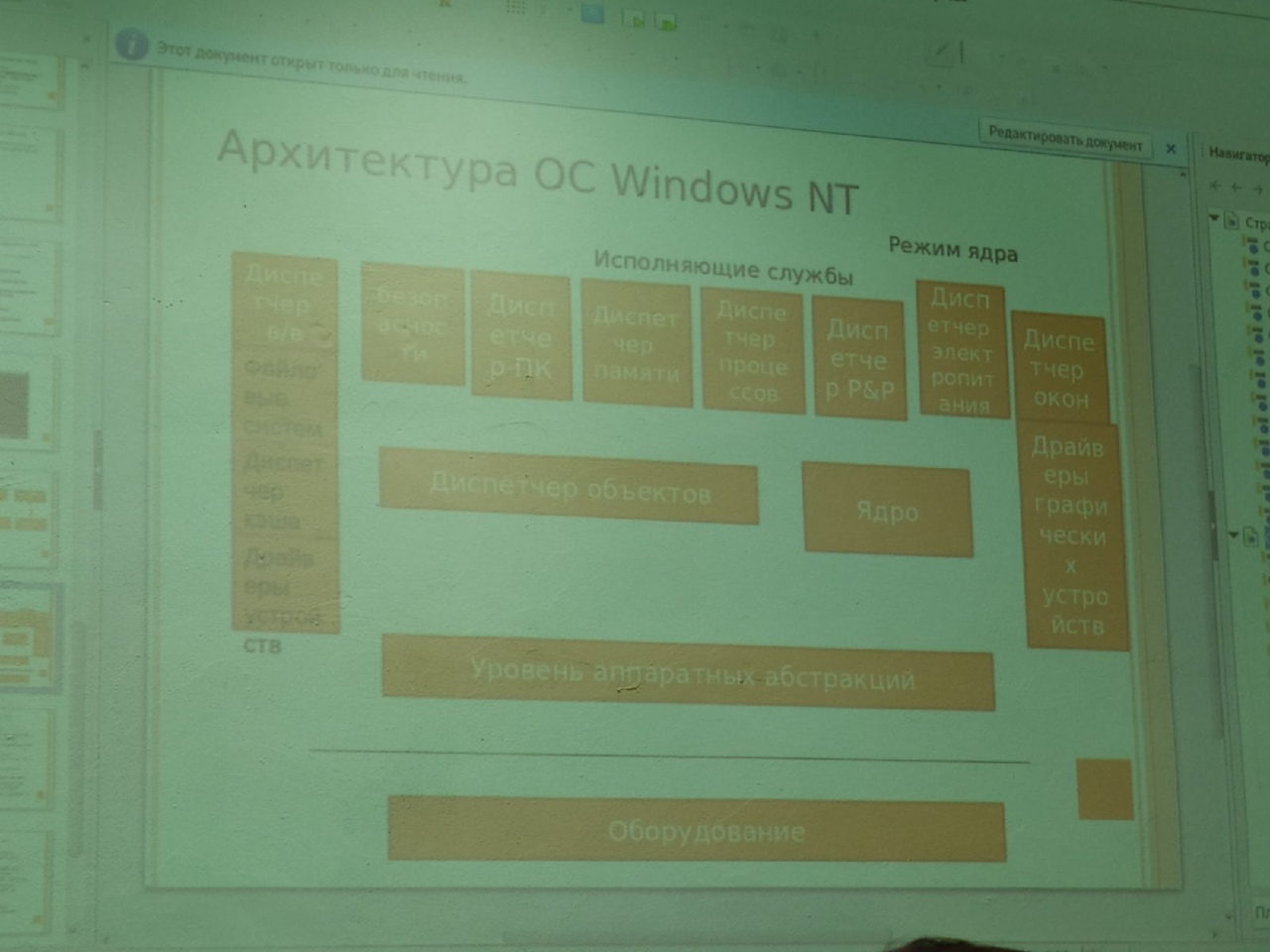
*Операционное окружение* – это среда, в которой пользователь запускает программу.

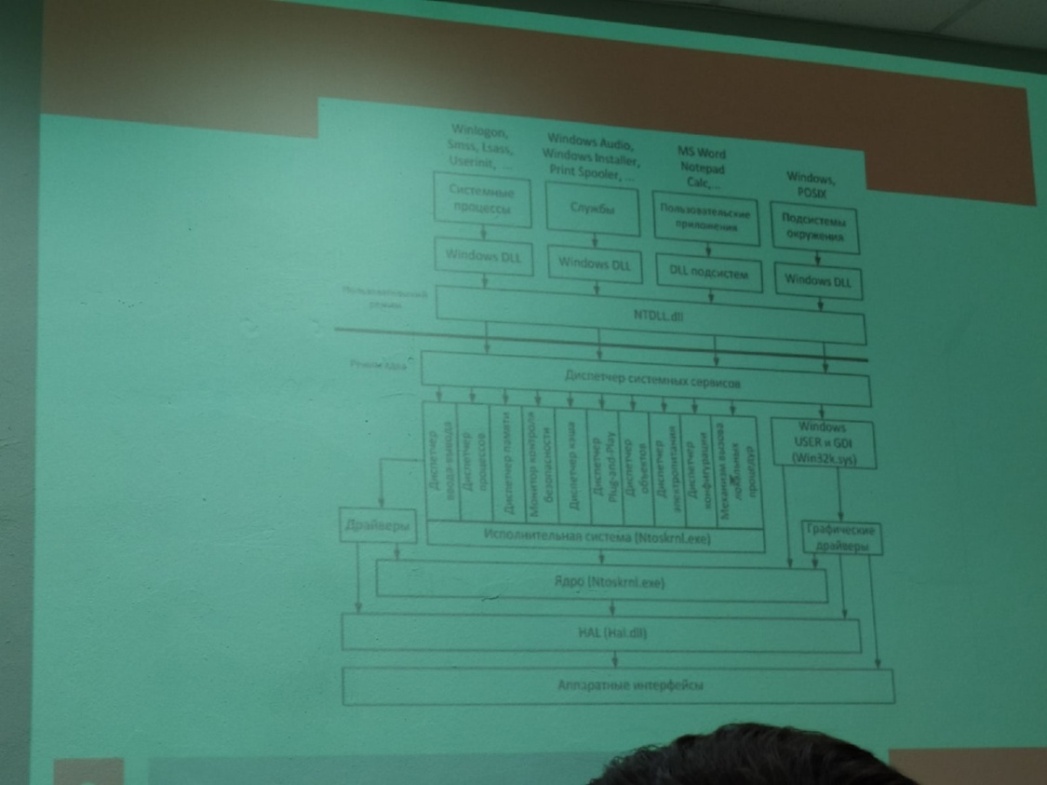
11.09.20

**Архитектура ОС Windows**

Режим пользователя:







В пользовательском режиме запускаются приложения пользователя, некоторые компоненты ОС. Это непривилегированный режим работы, приложение напрямую не могут обратиться к аппаратуре, только через исполняющие службы

Подсистема Win32позволяет запускать приложения 16, 32 разрядные приложения виндус и ДОС

POSIX приложение запускается для UNIX

Интегральная подсистема – управляет процессом регистрации

Режим ядра осуществляет доступ к данным системе и оборудования. Работает 3 службы: управляющие программы, ядро, уровень аппаратных абстракций

Ядро – планирует все системные операции, организует очереди данных, направляет их к микропроцессору и обрабатывает данные, отсылая их по соответствующему маршруту. Ядро планирует и управляет выполнением потоков согласно приоритетам, отвечает за синхронизацию нескольких процессов. Уровень аппаратных абстракций обеспечивает переносимость системы.

Диспетчер памяти отображает адреса виртуальной памяти в физические страницы ОЗУ. Диспетчер Plug and Playдля настройки параметров работы. Диспетчер электропитания контролирует питание на уровне ОС

Диспетчер окон и интерфейс графических устройств управляет системой окон, отвечает за отображение графики.

Диспетчер объектов реализует правила для сохранения, защиты и наименования объектов.

14.09.20

HAL – уровень аппаратных абстракций.

Фоновый режим – программы не требуют работы с пользователем. Работают на заднем фоне.

Структура Unix

Пользователи

Стандартные ПО (shell,компилятор, утилиты)

Стандартная библиотека (open, close, write, read, fork и тд)

ОС Unix (управление процессами, памятью, файловой системой, вводом/выводом)

Аппаратура (ЦП, память, диски, терминалы)

**Процессы**

*Процесс* – это минимальный программный объект, обладающий собственными системными ресурсами (запущенная программа, программа в стадии выполнения)

*Программа* – это план действий, а процесс-это само действие.

**Понятие процесса** включает:

* Программный код;
* Данные;
* Содержимое стека (*стек* – это среда для размещения данных для возврата из подпрограммы)
* Содержимое адресного и других регистров процессора.

Классификация процессов:

* По временным характеристикам: интерактивные, пакетные, реального времени
* По генеалогическому признаку:

Родительские и дочерние

* По времени развития:

Последовательные, параллельные (асинхронные, синхронные)

* По принадлежности к операционной системе: системные и пользовательские
* По связности: взаимосвязанные, изолированные, конкурирующие

Для управления процессами операционная система использует системные данные, которые существуют в течении всего времени выполнения процесса

Совокупность этих данных называет контекстом программы. Контекст определяет состояние процесса в данный момент времени. Для возобновления прерванного процесса необходимы системные данные (состояние регистров, программного счетчика и тд) Включает в себя пользовательский контекст (содержимое памяти процессора)

**Пользовательский контекст**: содержимое виртуального адресного пространства, сегментов программного кода, данных, стека, разделяемых сегментов и сегментов файлов, отображаемых в виртуальную память

**Регистровый контекст**: содержание аппаратных регистров: регистр счетчика команд, регистр состояния процессора, регистр указатель стека и регистры общего значения

*Дескриптор процесса* – системные данные, которые используются ОС для идентификации процесса.

Каждый процесс в ОС представлен набором данных называемых таблица управления процессом (ТУП), блок управления процессом (БУП)

В таблице находятся значения, которые характеризуют текущее состояние и используются операционной системой для управления прохождением процесса через компьютер.

Информация, хранящаяся в БУП:

* Текущее состояние процесса
* Уникальный идентификатор процесса
* Указатели памяти процесса
* Указатели выделенных ресурсов для процесса
* Область сохранения регистров, где хранится точка останов процессора
* Приоритет процесса

17.09.20

Операции над процессами:

* Создание процесса

1. Присвоение имени процессу
2. Включение этого процесса в список имен процессов, известных системе
3. Назначение начального приоритета процессу, формирование ТУП/БУП, выделение процессу начальных ресурсов

* Удаление процесса – удаление его из системы, при этом возвращаются ресурсы ОС, имя процесса удаляется из системных списков, таблица управления процессами освобождается и передаётся другому процессору
* Возобновление процесса – это операция по подготовке процесса начать выполнение с той точки, в которой он был прерван
* Изменение приоритета процесса
* Блокирование процесса

При запуске системы создаётся несколько процессов, некоторые из них являются высоко приоритетными, а некоторые являются фоновыми, они не связаны с конкретным пользователем, но выполняют особые функции. Они активизируются только в определенный момент по мере выполнения задач и называются – *демонами*

В unix-подобных операционных системах новый процесс создается с помощью системного вызова fork. Этот запрос создаёт дубликат вызываемого процесса, т.е. дочерний процесс является копией родительского, но отличаются индефикаторами. Затем дочерний процесс выполняет системный вызов для изменения образа памяти и запуска новой программы

В Windows для создания процесса и запуска программы используется CreateProcess. После создания нового процесса дочерние и родительские процессы имеют собственное адресное пространство, но дочерний процесс может использовать одинаковые ресурсы с родительским процессом.

Иерархия процессов (unix) - каждый новый созданный процесс является дочерним к предыдущему. Дочернему достаются от родительского достаются переменные, регистры. После вызова FORK, как только родительские данные скопированы, последующие изменения в одном из процессов не влияют на другой, но процессоры помнят о том, кто является родительским. Завершение процесса - используется системные вызовы exit (unix), а windows (exitprocess). Уничтожение процесса kill (unix), terminateprocess (windows)

После завершения освобождаются все системные ресурсы, завершаются дочерние процессы. Процесс может находится состояние *zombie*, если он не занимает процессорное время, не освобождает структуры ядра, в строке таблицы процессов он остаётся. Процесс может быть *сиротой*, если родительский процесс завершиться раньше дочернего, и он автоматически усыновляется процессом unit. Процесс может быть в состоянии сна, которое невозможно прервать, характеризуется тем, что не реагирует на системные вызовы, уничтожается только после перезагрузки системы.

События, приводящие к завершению процесса:

1. Плановые завершения
2. Выход по известной ошибке (отсутствие файла, данных)
3. Выход по неисправимой ошибке
4. Уничтожение другим процессом

Состояние процесса:

* Готовность
* Выполнение (активное состояние)
* Блокировка/ожидание

Понятие потоков

Процесс состоит из потоков.

*Поток* - последовательность команд процесса, которые выполняются независимо одна от другой и используются общие ресурсы одного процесса.

ОС являются многопоточными и многозадачными. Все потоки в рамках одного процесса имеют одно и то же адресное пространство.

Классификация потоков:

* По многозадачной модели

Существует вытесняющая многозадачность и не вытесняющая/кооперативная задачность

* По уровню реализации

**Невытесняющие алгоритмы** – активный поток выполняется до тех пор, пока он сам, по собственной инициативе, не отдаст управление ОС

**Вытесняющие алгоритмы** – решение о переключении процессора с одного потока на выполнение другого потока принимается ОС

**Алгоритмы планирования**: каждому потоку поочередно для выполнения предоставляется ограниченный непрерывный период процессорного времени – квант

Смена активного потока происходит если:

* Поток завершился и покинул систему
* Произошла ошибка
* Поток перешел в состояние ожидания
* Исчерпан квант процессорного времени

Планировщик создает 2 очереди готовых потоков:

1. 1 пришла в состояние готовности в результате исчерпания кванта
2. 2 – потоки завершили ввод/вывод

Приоритет – число, характеризующее степень привилегированности потока при использовании ресурсов вычислительной машины

Чем выше приоритет, тем выше привилегии. Чем выше привилегии потока, тем меньше времени он будет проводить в очередях.

Приоритет может изменяться в течение жизни потока:

* По инициативе самого потока
* Директивно администратором системы
* Назначается самой ОС – **динамические**
* **фиксированный**

Обслуживание с относительными приоритетами – выбирается поток с наивысшим приоритетом и активный поток выполняется до тех пор, пока он сам не покинет процесс

Обслуживание с абсолютными приоритетами – выбирается поток с наивысшим приоритетом

**Реализация потоков**

Есть 2 основных места реализации набора потоков пользовательском пространстве и в ядре

Реализация набора потоков в пользовательском пространстве. Весь набор потоков находится в одном пространстве. Об этом ядру ничего не известно

Использование потоков на уровне пользователя применимо для любой ОС

Недостатки: при выполнении ОС системного вызова блокируется не только данный поток, но и другие потоки данного процесса. Так как ядро закрепляется за каждым процессом только 1 процессор то несколько потоков 1 процесса не могут выполняться одновременно даже в случае многопроцессорной системы.

Реализация потоков в ядре: нет таблицы потоков в каждом процессе, у ядра есть таблица потоков в которой отслеживаются все потоки имеющиеся в системе

Недостаток: тратится время на переключение между потоками внутри одного процесса за счет переходов в режим ядра

Гибридная реализация. Часть потоков работает на уровне ядра, а часть на уровне пользователя

**Взаимодействие процессов**

Заключается в передаче данных между процессами или совместном использовании ресурсов

Синхронизация заключается в согласовании скоростей процессов путем престановки процесса до наступления некоторого события

**Виды межпроцессорного взаимодействия**

Процесс получатель и процесс отправитель

* Монолитные программы – содержат в своем коде все необходимые для работы инструкции. Обмен данными внутри таких программ производится при помощи передачи параметров функций и использование глобальных переменных. При запуске таких программ 1 процесс выполняет всю работу
* Модульные программы – состоит из отдельных модулей с четко определенными интерфейсами вызовов. Объединение модулей на этапе сборки исполняемого файла – СТАТИЧЕСКАЯ СБОРКА, либо на этапе выполнения программы – ДИНАМИЧЕСКАЯ СБОРКА.

Преимущества: один модуль можно заменить другим модулем; Модульная программа представляет 1 процесс, а данные передаются внутри процесса как параметры функции.

* Программы использующие межпроцессорное взаимодействие. Эти программы образуют программный комплекс, который решает общие задачи, каждая программа образует 1 или несколько процессов и каждый из процессов использует свои собственные данные и обменивается только результатом своей работы с другими пользователями, либо работают с общей областью данных, разделяемых между процессами. Процессы могут быть запущены на разных компьютерах и взаимодействуют через сеть

Механизмы взаимодействия должны быть заложены в ОС, а не в прикладные программы.

Механизмы процессорного

7 Механизмов:

* Механизм прерывания – реакция на прерывание заключается в выполнении программного кода, который находится в памяти, адресуемые ОС. ОС поддерживает специальную область памяти – таблицу прерываний, где каждому прерыванию ставится соответствие адрес памяти, по которому находится программный код – обработчик прерываний. ОС поддерживает от 16 до 256 прерываний. После выполнения обработчика ОС возвращает управление задачам, которые были активны до поступления прерывания.
* Сигналы – механизм передачи требований от одного процесса другому не немедленное выполнение действия. Обработчик сигнала создается процессом и помещается в начале процесса. ОС поддерживает таблицу обработчиков сигнала для каждого процесса. В ней каждому сигналу ставится в соответствии адрес обработчика сигнала. Обработчик сигнала хранится в памяти процесса.
* Сообщение – используется для обмена данными между процессами. Для этого создается специальная очередь сообщений, которая поддерживается ОС. В очереди накапливаются данные, которые записываются в нее процессами. Накопленные данные могут быть считаны другими процессами. ОС поддерживает специальную область оперативной памяти, где хранятся очереди. Первый процесс должен создать очередь, а остальные процессы должны получить доступ к ней.
* Именованные каналы – используются для обмена данными между процессами. В качестве очереди используется файл специального вида. Процессы могут записывать в этот файл и считывать из него данные. Все процессы имеют равноправный доступ к именованному каналу.
* Гнезда или сокеты – используются для взаимодействия процессов, выполняемых на разных компьютерах. Каждый процесс создает гнездо, в которое записывает данные, и считывает их из него. Гнезда связаны сетевыми протоколами (TCP/IP).
* Общая память – механизм общей памяти заключается в том, что в адресное пространство процессов отображается один и тот же участок физической памяти, адресуемой ОС.

*Критическая секция* – часть программы, результат выполнения которой может меняться если переменные, относящиеся к этой части программы, изменяются другими процессами в то время, пока выполнение этой части еще не завершено.

*Существует правило нахождения критической секции –* процесс не может быть заблокирован на этом участке, он должен как можно скорее его пройти.

* Семафоры – используется для предотвращения конфликтной ситуации при одновременном доступе к общей памяти. Это специальные флаги, которые указывают на возможность использования того или иного участка памяти (ставится временный флаг на участок). Ситуация, когда и более процесса обрабатывают разделяемые данные и конечный результат зависит от скорости процесса, называется *гонками.*

25.09.2020

**Планирование процессов**

Планирование – распределение ресурсов между имеющимися процессами и наоборот

Уровни планирования:

* Долгосрочное – планировщик решает, какой из процессов, находящиеся во входной очереди (во внешней памяти), должен быть переведен в очередь готовых процессов в случае освобождения ресурсов памяти
* Краткосрочное – планировщик решает, какой из процессов, находящихся в очереди готовых процессов, должен быть передан на выполнение процессору
* Среднесрочное (дополнительное)

Стратегии планирования:

* Первый пришел – первый обслуживается FIFO – first come – first served (FCFS) – CPU передается тому процессу, который раньше всех других его запросил. Среднее время ожидания для этой стратегии – большое, и зависит от порядка поступления процессов в очередь готовых процессов. Недостаток: «эффект конвоя» - бесконечная очередь
* Короткая работа выполняется первой – SJF – Shortest Job First. Недостаток: невозможно оценить величину времени обслуживание.

Эти две стратегии используются в системах пакетной обработки, где есть долгосрочный планировщик.

* Приоритетное планирование – каждому процессу приписывается приоритет, определяющий очередность предоставления ему процессора.

Недостаток: блокирование низкоприоритетных процессов на неопределенно долгое время.

* «Карусельная» (циклическая) RR - Round Robin (применяется в системах разделения времени). Очередь готовых процессов рассматривается как кольцевая, процессы циклически перемещаются по очереди, получая процессор на время равный кванту. Новый процесс добавляется в хвост очереди, если процесс не закончил свою работу в пределах выделенного кванта времени – его работа принудительно прерывается, и он перемещается в хвост очереди.
* Планирование с использованием многоуровневой очереди. В этом случае очередь готовых процессов разделяется на несколько очередей, в каждой очереди находятся процессы с одинаковыми свойствами и каждый из который может планироваться индивидуальной стратегией. Взаимодействие осуществляется следующим образом – ни один процесс с более низким приоритетом не может быть запущен пока не выполнятся все процессы во всех очередях с более высоким приоритетом.
* Многоуровневая очередь с обратными связями – тоже самое, но процессы могут перемещаться между очередями. Все новые процессы поступают в конец первой очереди, первой запрос из n очереди поступает на обслуживание лишь тогда, когда все очереди от первой до n-1 пустые. На обслуживание выделяется квант времени.

Планирование в системах реального времени – в мягких системах можно использовать те же стратегии, что и в интерактивных системах, а в системах жестких создается расписание.

Функции ОС по управлению памяти

* Отслеживание свободной и занятой памяти
* Выделение памяти процесса и освобождение ее при завершении процессов
* Защита адресных пространств процессов
* Организация виртуальной памяти
* Настройка адресов программы на конкретную область физической памяти

Часть ОС, которая управляет памятью, называется менеджером памяти

Управление памяти – распределение ее между процессами

Типы адресов:

* Символьные имена – присваивает пользователь при написании программы
* Виртуальные адреса – вырабатывает транслятор, переводящий программу на машинный язык

Совокупность виртуальных адресов процесса называют виртуальным адресным пространством. Максимальный размер определяется разрядностью процессора

* Физические адреса – соответствую номерам ячеек оперативной памяти, где в действительности будут расположены переменные и команды

Каждый процесс имеет свое адресное пространство. Способы от виртуальных адресов к физическим осуществляется двумя способами:

* Перемещающий загрузчик на основании имеющихся у него исходных данных о начальном адресе физической памяти и информации, которую предоставляет транслятор выполняет загрузку программы, совмещая ее с заменых виртуальных адресов физическими.

Недостаток: программа жестко привязывается к выделенному участку памяти

* Программа загружается в память в неизменном виде в виртуальные адреса, но ОС фиксирует смещение действительного расположения программного кода относительного виртуального адресного пространства.

Более гибкий и применяется в современных вычислительных системах

**Алгоритмы распределения памяти**

* Распределение памяти фиксированными разделами – оперативная память разделяется на несколько разделов фиксированной величины. Связывание логических адресов процесса и физических происходит на этапе его загрузки в конкретный раздел

Стратегии загрузки:

* Стратегия первого подходящего – задание помещается в первый подходящий по размеру раздел
* Наиболее подходящий – помещается в тот раздел, где ему наиболее тесно
* Наименее подходящий участок – выбирается самый большой раздел
* Следующий подходящий участок – выбирается раздел, который следует за каким-либо подходящим участком.

Недостаток распределение по разделам: фрагментация – наличие большого числа несмежных участков свободной памяти

* Оверлейная структура – используется структура с перекрытием, что значит держать в памяти те участки программы, которые нужны только в данный момент. Размер виртуального адресного пространства больше, чем имеющаяся память.
* Распределение памяти динамическими разделам – оперативная память делится на разделы, но границы разделов не фиксируются.

Недостаток – фрагментация.

* Перемещаемый раздел (сжатие) – способ борьбы с фрагментацией. Сжатие – перемещение всех занятых участков в сторону старших или в сторону младших адресов, чтобы свободная память образовывала единую свободную область.

Недостатки – все процессы переходят в состояние ожидания, расходуется процессорное время.

* Защита памяти –организована при помощи одного граничного регистра, содержащего адрес границы ОС

**Виртуальная память**

Swapping – образы процессов выгружаются на диск и возвращаются в оперативную память целиком.

Недостатки:

* Медленная работа системы, потому что весь процесс выгружается
* Неспособность загрузить для выполнения процесс, если виртуальное адресное пространств превышает имеющуюся свободную память
* Неэффективное использование памяти

Virtual memory – между оперативной памятью и диском перемещаются части образов процесса

Модели:

* Страничные
* Сегментные
* Сегмента-страничные

Страничная виртуальная память – организует перемещение данных между памятью и диском страницами, частями виртуального адресного пространства фиксированного и небольшого размера.

Виртуальные страницы – виртуальное адресное пространство каждого процесса делится на части одинакового фиксированного размера

Вся оперативная память делится на части такого же размера – это физические страницы.

Страницы имеют фиксированную длину, обычно это 2k и не могут перекрываться

При создании процесса ОС загружает в оперативную память несколько виртуальных страниц

Для каждого процесса ОС создает таблицу страниц

Запись таблицы называется дескриптором страницы и включает информацию:

* Номер физической страницы – место, куда загружена виртуальная страница
* Признак присутствия – если страница находится в оперативной памяти - 1, если нет – 0
* Признак модификации страницы – если страница изменялась – 1, если нет – 0
* Признак обращения к странице – если происходит обращение к адресу – 1, если нет – 0

Сами таблицы размещаются в оперативно памяти, а адрес таблицы страницы – в специальном регистре процессора

Если нужная виртуальная страница находится в оперативной памяти, то выполняется преобразование виртуального адреса в физический, то есть виртуальный адрес заменяется указанным в записи таблицы физическим адресом.

Если нужная виртуальна страница выгружена на диск, то происходит страничное прерывание. Выполняющийся процесс переводится в состояние ожидания и активизируется другой процесс из очереди готовых.

Программа по обработке страничного прерывания находит на диске нужную виртуальную страницу и пытается загрузить ее в оперативную память

Если в памяти есть свободная физическая страница, то загрузка выполняется немедленно, если же нет свободны страниц, то необходимо какую-то выгрузить из оперативной памяти

Стратегии выгрузки страниц:

1 алгоритм – LRU. Дольше всего неиспользовавшаяся страница. Делается вывод, что она не будет нужна в ближайшее время

2 алгоритм – первая попавшаяся страница (случайный выбор)

3 алгоритм – NRU. Станица, к которой в последнее время было меньше всего обращений

4 алгоритм – FIFO. Выгружается та страница, которая дольше всех находилась в оперативной памяти

5 алгоритм – кольцевой. Напоминает предыдущий алгоритм, но очередь образует кольцевой список

Свойства страничной организации:

* Объем страница равен 2k
* Смещение в виртуальном и физических адресах равны между собой

Сегментное распределение виртуальной памяти

Сегментное распределение предусматривает перемещение данных сегментами, частями виртуального адресного пространства произвольного размера с учетом смыслового значения

Адресация при сегментной организации виртуальной памяти осуществляется с помощью двух значений: номер сегмента и смещение внутри сегмента

С каждым сегментом ассоциируется его дескриптор, который содержит информацию о разделе и атрибутах сегмента

На этапе создания процесса во время загрузки его в оперативную память создается таблица сегментов, в которой указывается базовый физический адрес сегмента в оперативной памяти, размер сегмента, правила доступа к сегменту, признаки модификации, присутствия и обращения к данному сегменту.

Если во время выполнения процесса происходит обращение по виртуальному адресу относящемуся сегменту, который в данный момент отсутствует в памяти – происходит прерывание

Недостаток – операция адресации в этой модели медленная

Сегментно-страничное распределение виртуальной памяти

Виртуальное адресное пространство процесса делится на сегменты. Каждый виртуальный сегмент делится на страница равного размера.

В современных ОС используется страничное управление виртуально памятью

Операционная система Linux:

**32x.** 4гб памяти. 3гб – пользовательская часть, 1гб – системная часть

**64x.** Пользовательская – до 512гб, остальное под системную область

В ОС семейства Unix в качестве виртуальной памяти используется специальный дисковый раздел. Выгружается та память, которая дольше всех не использовалась.

Операционная система Windows:

**32x.** 4гб памяти. 2гб – пользовательская часть, 2гб – системная часть

**64x.** 16Тб памяти. 8Тб – пользовательская часть, 8Тб – системная часть

Используется файл подкачки. Он может изменяться динамически, в зависимости от использования виртуальный памяти. Размер файла подкачки должен быть не менее в 2,5 раза больше, чем размер оперативной памяти.

Файловая система позволяет организовать размещение данных на диске.

Компоненты файловой системы:

* Все файлы на диске
* Набор данных для управления файлами
* Набор программных средств, которые позволяют выполнять различные операции над этими данными

Функции файловой системы:

* связать имя файла с выделенным ему пространстве во внешней памяти
* Распределение внешней памяти между файлами
* Защиты несанкционированного доступа
* Обеспечение совместного доступа к файлам

Файловая система имеет иерархическую структуру – создаётся несколько уровней, причем нижние уровни входят в каталог высокого уровня, ну тип папочки, где d/user/xyuser

Один из видов файловых систем – таблица размещения файлов. (fat12, fat16, fat32)

**Организация хранения данных на диске**

Файл – пак связанной между собой информации на диске или другом носителе. В этот пак можно как записывать, так и считывать данные. Эти файлы хранятся во внешней памяти и им пофиг на питание.

Файлы используют для того, чтобы что-то долго хранить, или же чтобы совместно использовать

Бывают **обычные файлы** и **специальные**

Обычные файлы включают в себя рандомную инфу, ее может заносить пользователь, или программа, которая работает.

Специальные файлы пахают на благо устройств ввода – вывода, они также позволяют пользователю выполнять эти операции ввода вывода.

Каталог – дофига особый файл, он содержит всю системную инфу о разных файлах, которые группируются по какому-либо признаку

Эти каталоги содержат списки файликов, еще они смотрят на характеристики и на сам файл, и находят пары)

**Имена файлов**

Простое имя – может определять всякую штуку только в одном каталоге, его присваивает пользователь и ОС может запрещать некоторые символы

Полное имя – это штука, в который пишется весь путь до файла, от корня

Относительное имя – это имя только для каталога, в котором в данный момент работает пользователь \*текущий каталог

Правила наименования зависят от самой ОС. Большинство позволяют задавать как само имя, так и расширение. Расширение файла позволяет оформить работу с другими штуками. И в именах могут быть символы ? и \*.

**Атрибуты файла**

Каждый файл также связывают со всякими штуками, например, размер и дата. Эти штуки и есть атрибуты, они же свойства файла. У каждой ОС разный список атрибутов. Но они обязательно должны включать:

* Основную информацию (тип, имя)
* Адресная информация (начальный адрес, размер)
* Информация об управлении доступом (владелец, допустимые операции)
* Информация об использовании (дата создания, дата последнего чтения, дата модификации)

**Доступ к файликам**

* **Последовательный** - процесс читает или записывает всякие штуки только последовательно, тобиш от начал к концу.
* Прочитаны же они могут быть в произвольном порядке – это **произвольный доступ** к файлам

**Операции над файлами**

* Create. Создание файла, не содержащего данных.
* Open. Перед использованием файла процесс должен его открыть
* Close. Если работа с файлом завершена, его атрибуты и адреса блоков на диске больше не нужны.
* Read. Чтение данных из файла.
* Write. Запись данных в файл с текущей позиции
* Get attributes. Предоставляет процессам нужные им сведения об атрибутах файла.
* Set attributes – установить атрибуты.
* Rename. Возможность переименования файла.
* Delete. Удаление файла.

**Действия, которые ОС выполняет над файлами**

* По его символьному имени может узнать его характеристики
* Потом скопировать эти характеристики в оперативку, это вроде как программный код
* Из того, что она узнает, она может давать права пользователя
* Может очищать память, выделенную под временное хранение этих характеристик
* И также проводить всякие уникальные шутки над файлами

**Организация хранения данных на диске**

**1 способ**. Все пространство делится на блоки – они же кластеры. Кластеры имеют свои номерки и там хранятся всякие данные пользователя или же служебная информация для сборки всяких данных.

Недостаток этого способа – список номер всех кластеров сам занимает кластеры, и файлы получаются очень ограниченные

**2 способ.** Наборы данных размещаются в соседних кластерах, а в служебном блоке хранится номера первого и последнего кластера

**3 способ.** Каждый блок имеет в себе маленькую область. И в этой частичке хранится номер следующего блока. И благодаря этому образуется линейный список, а в служебном блоке хранится номер первого блока.

**Дорожки, цилиндр и сектора**

На каждой стороне пластинок (что такое пластинка\*??) есть тоненькие кольца, они же **дорожки**, и на них хранятся данные

Совокупность этих дорожек одного размера на всех пластинках называется cylinder. Каждая де дорожка делится на фрагменты, и при этом все дорожки имеют равное число этих фрагментов. В один сектор (фрагмент) можно записать одно и тоже число байт (512 байт).

Сектор – наименьшая единица, которая адресуется (передается) как данные между дисковым пространством и оперативкой. Также сектор - это участок магнитного диска, который может как считан с этого диска, так и записан на него за один прием.

**Кластеры**

Кластер – это минимальная единичка памяти диска, которая дается файлу

Каждый файл занимает цельное число кластеров. И при этом каждый файл — это смежное число секторов. Размер же кластера зависит как от файловой системы, так и от размера диска.

Дорожки и секторы создаются в результате форматирования диска. Форматирование бывает физическое или низкоуровневое.

Низкоуровневый формат диска не зависит от типа ОС, которая использует его.

Разметку диска под разные типы Ф.С выполняют процедуры высокоуровневого форматирования. При этом определяется размер кластера и на диск записывается информация, необходимая для работы Ф.С.