**Вопросы экзамена**

1. Определение операционной системы

**Операционная система** - это единственная программа, которая выполняется на вычислительной системе с момента включения (вычислительной системы) и до её выключения и которая выполняет следующие основные функции:

предоставляет процессам «абстрактную» вычислительную машину,

управляет ресурсами (доступом к ресурсам),

организует многозадачную среду и разграничивает процессы (пользователей) друг от друга,

учитывает процессы, ресурсы и пользователей.

1. Эволюция операционных систем: первый и второй периоды.
2. Первый период (1945 - 1955). Ламповые машины. Операционные системы отсутствуют.

Программирование осуществлялось исключительно на машинном языке.

Операционные системы отсутствуют.

Все задачи организации вычислительного процесса решались вручную каждым программистом с пульта управления. За пультом мог находиться только один пользователь. Вычислительная система выполняла одновременно только одну операцию (ввод-вывод, собственно вычисления, размышления программиста).

В конце этого периода появляется первое системное программное обеспечение:

* в 1951-52 гг. возникают прообразы первых компиляторов с символических языков (Fortran и др.),
* в 1954 г. Nat Rochester разрабатывает ассемблер для IBM-701.

2. Второй период (1955 - Начало 60-х). Компьютеры на основе транзисторов. Пакетные операционные системы.

Применение транзисторов привело к повышению надежности компьютеров.

Эксплуатация и обслуживание вычислительной техники подешевели.

Началось использование ЭВМ коммерческими фирмами.

Наблюдается бурное развитие алгоритмических языков (ALGOL-58, LISP, COBOL, ALGOL-60, PL-1 и т.д.). Появляются первые настоящие компиляторы, библиотеки математических и служебных подпрограмм. Упрощается процесс программирования.

Происходит разделение персонала на программистов и операторов, специалистов по эксплуатации и разработчиков вычислительных машин.

Изменяется сам процесс прогона программ:

1. Пользователь приносит программу с входными данными в виде колоды перфокарт и указывает требуемые для нее ресурсы.
2. Полученные выходные данные печатаются на принтере, и пользователь получает их обратно через некоторое (довольно большое) время.

Смена запрошенных ресурсов вызывает приостановку выполнения программ. В результате процессор часто простаивает.

Для повышения эффективности использования компьютера задания с похожими требуемыми ресурсами начинают собирать вместе, создавая пакет заданий. Появляются первые системы пакетной обработки, которые просто автоматизируют запуск одной программы из пакета за другой и, тем самым, увеличивают коэффициент загрузки процессора.

При реализации систем пакетной обработки был разработан формализованный язык управления заданиями, с помощью которого программист сообщал системе и оператору, какую работу он хочет выполнить на вычислительной машине.

**Системы пакетной обработки** явились прообразом современных операционных систем, они стали первыми системными программами, предназначенными для управления вычислительным процессом.

1. Эволюция операционных систем: третий и четвертый периоды.
2. Третий период (Начало 60-х - 1980). Компьютеры на основе интегральных микросхем. Первые многозадачные ОС.

Вычислительная техника становится более надежной и дешевой.

Растет сложность и количество задач, решаемых компьютерами.

Повышается производительность процессоров.

Повышению эффективности использования процессорного времени мешает низкая скорость механических устройств ввода-вывода.

Появление магнитного диска, для которого не важен порядок чтения информации, то есть устройства прямого доступа, привело к дальнейшему развитию вычислительных систем.

Введение техники подкачки-откачки данных в пакетные системы позволило совместить реальные операции ввода-вывода одного задания с выполнением другого задания, но потребовало появления аппарата прерываний для извещения процессора об окончании этих операций.

Дальнейшее повышение эффективности использования процессора было достигнуто с помощью мультипрограммирования.

1. Четвертый период (1980 - настоящее время). Персональные компьютеры. Классические, сетевые и распределенные системы.

Связан с появлением больших интегральных схем (БИС).

Компьютер по цене и простоте эксплуатации стал доступен отдельному человеку, а не отделу предприятия или университета. Наступила **эра персональных компьютеров**.

Компьютеры стали широко использоваться неспециалистами, что потребовало разработки "дружественного" программного обеспечения, это положило конец кастовости программистов.

Бурно развиваются сети компьютеров работающих под управлением сетевых или распределенных операционных систем.

**Распределенная система** внешне выглядит как обычная автономная система.

1. Основные функции операционных систем

1. Планирование заданий и использования процессора:

* Обеспечение режима мультипрограммирования, то есть организация параллельного выполнения двух или более программ на одном процессоре, создающая видимость их одновременного исполнения.
* Планирование и диспетчеризация задач в соответствии с заданными стратегией и дисциплинами обслуживания.
* Обеспечение работы систем программирования, с помощью которых пользователи готовят свои программы.

2. Обеспечение программ средствами коммуникации и синхронизации:

* Запуск программы (передача ей управления, в результате чего процессор исполняет программу).
* Идентификация всех программ и данных.
* Прием и исполнение различных запросов от выполняющихся приложений. ОС умеет выполнять очень большое количество системных функций (сервисов), которые могут быть запрошены из выполняющейся программы. Обращение к этим сервисам осуществляется по соответствующим правилам, которые и определяют интерфейс прикладного программирования (API) этой ОС.
* Организация механизмов обмена сообщениями и данными между выполняющимися программами.
* Для сетевых ОС характерной является функция обеспечения взаимодействия связанных между собой компьютеров.

3. Управление памятью:

* Загрузка в оперативную память подлежащих исполнению программ.
* Распределение памяти, а в большинстве современных систем и организация виртуальной памяти.

4. Управление файловой системой:

* Обслуживание всех операций ввода-вывода.
* Обеспечение работы систем управлений файлами и/или систем управления базами данных, что позволяет резко увеличить эффективность всего программного обеспечения.

5. Управление вводом-выводом:

* Прием от пользователя (или от оператора системы) заданий, или команд, сформулированных на соответствующем языке, и их обработка. Задания могут передаваться в виде текстовых директив (команд) оператора или в форме указаний, выполняемых с помощью манипулятора (например, с помощью мыши). Эти команды связаны, прежде всего, с запуском (приостановкой, остановкой) программ, с операциями над файлами (получить перечень файлов в текущем каталоге, создать, переименовать, скопировать, переместить тот или иной файл и др.), хотя имеются и иные команды.

6. Обеспечение безопасности:

* Защита одной программы от влияния другой, обеспечение сохранности данных, защита самой ОС от исполняющихся на компьютере приложений.
* Аутентификация и авторизация пользователей (для большинства диалоговых операционных систем). Под аутентификацией понимается процедура проверки имени пользователя и его пароля на соответствие тем значениям, которые хранятся в его учетной записи. Очевидно, что если входное имя (login2) пользователя и его пароль совпадают, то, скорее всего, это и будет тот самый пользователь. Термин авторизация означает, что в соответствии с учетной записью пользователя, который прошел аутентификацию, ему (и всем запросам, которые будут идти к операционной системе от его имени) назначаются определенные права (привилегии), определяющие, что он может, а что не может делать на компьютере.
* Удовлетворение жестким ограничениям на время ответа в режиме реального времени (характерно для операционных систем реального времени).
* Предоставление услуг на случай частичного сбоя системы. Операционная система изолирует аппаратное обеспечение компьютера от прикладных программ пользователей. И пользователь, и его программы взаимодействуют с компьютером через интерфейсы операционной системы.

1. Типы архитектур операционных систем: монолитная система

Монолитная система

Компоненты операционной системы являются не самостоятельными модулями (программами), а составными частями одной большой программы.

Монолитное ядро это такая схема операционной системы, при которой все ее компоненты являются составными частями одной программы, используют общие структуры данных и взаимодействуют друг с другом путем непосредственного вызова функций.

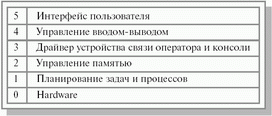
Для монолитной операционной системы ядро операционной системы совпадает со всей операционной системой.

1. Типы архитектур операционных систем: слоеная система

Слоёная система

Вся вычислительная система делится на ряд более мелких уровней с хорошо определенными связями между ними, так чтобы объекты уровня N могли вызывать только объекты из уровня N-1.

Нижним уровнем в таких системах обычно является hardware, верхним уровнем - интерфейс пользователя (интерфейс прикладного программирования - API).

Чем ниже уровень, тем более привилегированные команды и действия может выполнять модуль, находящийся на этом уровне.

1. Типы архитектур операционных систем: микроядерная система

Микроядерная система

Модель клиент-сервер - это еще один подход к структурированию ОС. В широком смысле модель клиент-сервер предполагает наличие программного компонента - потребителя какого-либо сервиса - клиента, и программного компонента - поставщика этого сервиса - сервера. Взаимодействие между клиентом и сервером стандартизуется, так что сервер может обслуживать клиентов, реализованных различными способами и, может быть, разными производителями. При этом главным требованием является то, чтобы они запрашивали услуги сервера понятным ему способом. Инициатором обмена обычно является клиент, который посылает запрос на обслуживание серверу, находящемуся в состоянии ожидания запроса. Один и тот же программный компонент может быть клиентом по отношению к одному виду услуг, и сервером для другого вида услуг.

Применительно к структурированию ОС идея состоит в разбиении ее на несколько процессов - серверов, каждый из которых выполняет отдельный набор сервисных функций - например, управление памятью, создание или планирование процессов. Каждый сервер выполняется в пользовательском режиме. Клиент, которым может быть либо другой компонент ОС, либо прикладная программа, запрашивает сервис, посылая сообщение на сервер.

Разработка операционных систем с перенесением значительной части системного кода на уровень пользователя и одновременная минимизация ядра. В этом случае большинство ее составляющих являются самостоятельными программами и взаимодействие между ними обеспечивает специальный модуль ядра, называемый микроядром.

Микроядро работает в привилегированном режиме и обеспечивает взаимодействие между программами, планирование использования процессора, первичную обработку прерываний, операции ввода-вывода и базовое управление памятью. Остальные компоненты системы взаимодействуют друг с другом путем передачи сообщений через микроядро.

Главный принцип разделения работы между микроядром и окружающими его модулями - включать в микроядро только те функции, которым абсолютно необходимо исполняться в режиме супервизора и в привилегированном пространстве.

1. Типы архитектур операционных систем: смешанная (гибридная) система

Смешанные (гибридные) системы

Примером этого подхода является операционная система Linux. Она имеет монолитно-слоёно-модульную архитектуру.

Существует монолитное слоёное ядро, включающее основные модули системы:

- «Управление процессами»,

- «Управление памятью»,

- «Виртуальная файловая система»,

- «Подсистема ввода-вывода»,

- «Сетевая подсистема»,

- «Межпроцессное взаимодействие».

В ней также присутствует очень большое количество (сотни) элементов (прежде всего, драйверы файловых систем и устройств), которые оформлены как внешние модули и которые могут подгружаться ядром по мере необходимости.

При этом, все компоненты ядра, в том числе подгружаемые модули, работают в одном адресном пространстве и активно используют общие структуры данных, что свойственно операционным системам с монолитным ядром.

1. Архитектура операционной системы Linux.

Подобное разбиение вычислительной системы соответствует «слоёному» типу построения вычислительных систем: каждый слой (подсистема) может взаимодействовать лишь с соседними слоями.

Кроме того, зависимости между подсистемами организованы по принципу «сверху вниз» («интерфейсный» тип): слои, изображённые ближе к вершине, зависят от нижних слоёв, в то время как расположенные ближе к основанию не зависят от верхних.

Ядро Linux состоит из пяти основных подсистем:

- планировщик процессов,

- менеджер памяти,

- виртуальная файловая система,

- сетевой интерфейс,

- подсистема межпроцессного взаимодействия.

* Планировщик процессов использует менеджер памяти для настройки схемы распределения аппаратной памяти для определенного процесса, когда процесс этот возобновлен.
* Подсистема межпроцессного взаимодействия использует менеджер памяти, чтобы поддерживать механизм связи с совместно используемой памятью. Этот механизм позволяет двум процессам иметь доступ к некой общей области памяти (не считая того, что каждый из процессов имеет и собственную память).
* Виртуальная файловая система использует сетевой интерфейс для поддержки сетевой файловой системы (NFS), а также использует менеджер памяти для предоставления такого устройства, как RAM-диск (программная технология, позволяющая хранить данные в быстродействующей оперативной памяти как на блочном устройстве).
* Менеджер памяти использует виртуальную файловую систему для поддержки подкачки (swapping) или перемещения страниц (paging); это единственная причина, по которой менеджер памяти зависит от планировщика процессов. Когда процесс хочет получить доступ к содержимому памяти, которое в данный момент времени сохранено (swapped out) в некотором постоянном хранилище (как правило, в специальном swap-разделе жесткого диска), менеджер памяти обращается с запросом к файловой системе достать это содержимое из хранилища и приостанавливает процесс, пока запрос не будет завершён.

1. Пользователи операционной системы

Операционная система (ОС) - программный комплекс, предоставляющий пользователю среду для выполнения прикладных программ и управления ими, а прикладным программам средства доступа и управления аппаратными ресурсами.

Каждый пользователь ОС использует в своей деятельности инструменты, предоставляемые либо непосредственно ядром ОС, либо работающими под управлением ОС прикладными программами.

Для решения своих задач пользователь формализует описание задачи на некотором входном языке для ОС или программ.

Синтаксис и семантика таких языков различна в зависимости от решаемых с их помощью задач.

Пользовательские задачи могут быть разделены на следующие группы:

- расширение функциональности ОС;

- конфигурирование режимов работы ОС;

- разработка прикладных программ;

- решение прикладных задач при помощи готовых программ.

Пользователей ОС, применяющих тот или иной язык общения с ОС, можно подразделить на несколько групп:

- системные программисты;

- системные администраторы;

- прикладные программисты;

- прикладные пользователи.

|  |  |  |
| --- | --- | --- |
| **Роль пользователя** | **Решаемая задача** | **Входной язык** |
| **Системный программист** | Расширение функций ОС | Низкоуровневые языки разработки, в том числе Ассемблер |
| **Системный администратор** | Конфигурирование ОС и регистрация пользователей | Форматы конфигурационных файлов и языки управления средствами администрирования |
| **Оператор** | Текущее администрирование системы, установка/удаление программного обеспечения, его настройка | Форматы конфигурационных файлов инсталляторов и устанавливаемого программного обеспечения |
| **Специалист по аппаратному обеспечению** | Обслуживание аппаратуры, ввод ее в эксплуатацию, вывод из эксплуатации | Языки средств настройки оборудования для использования в конкретной ОС |
| **Прикладной программист** | Разработка программного обеспечения, предназначенного для решения задач прикладного пользователя | Языки высокого уровня, интерфейс системных вызовов ядра ОС |
| **Администратор данных** | Архивирование данных системы, управление информационными ресурсами (базы данных, справочники) | Языки управления и конфигурирования используемых программных средств |
| **Прикладной пользователь** | Решение конкретных прикладных задач при помощи готового программного обеспечения | Языки управления заданиями ОС, языки управления используемыми программными средствами |

1. Понятие интерфейса операционной системы

Определение 1.

Интерфейс - взаимодействие между объектами по вертикали, как вышестоящего с нижестоящим, как старшего с младшим.

Определение 2.

Интерфейс - совокупность аппаратно-программного обеспечения и документации, определяющего взаимодействие между вышестоящими и нижестоящими уровнями в одной системе.

Для внешней среды (пользовательским процессам) ядро операционной системы предоставляет интерфейс виртуальной машины.

Пользовательские процессы (системное и прикладное программное обеспечение) пишутся (программируются) без каких-либо знаний о физическом оборудовании, имеющемся на данном компьютере: ядро абстрагирует все особенности оборудования в четком виртуальном интерфейсе - API (Application Programming Interface - интерфейс прикладного программирования).

1. Интерфейсы операционной системы: системные вызовы

Интерфейс прикладного программирования API (Application Programming Interface) - это некоторый механизм, который позволяет пользовательским программам обращаться за услугами ядра операционной системы. Этот механизм состоит из системных вызовов.

Пользовательская программа запрашивает сервис у операционной системы, осуществляя системный вызов. Физически системный вызов представляет собой имя функции ядра, видимое извне, из-за пределов операционной системы.

Зная имя функции, что делает функция, и какие параметры обрабатывает функция, можно вызвать функцию, передав ей нужные данные, и тем самым, заставить её что-то сделать. В этом смысл системного вызова.

Однако, это же чужая функция, функция другой программы. А вызвать из некоторой программы функцию другой программы невозможно.

Но в данном случае, этой другой программой является сама операционная система - программа, которая полностью контролирует оборудование.

1. Интерфейсы операционной системы: прерывания

Прерывание (interrupt) - событие, генерируемое внешним (по отношению к процессору) устройством.

Посредством **аппаратных прерываний** (hardware interrupt) 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).

Каждый тип аппаратных прерываний имеет собственный номер, однозначно определяющий источник прерывания.

Аппаратное прерывание - это асинхронное событие, то есть оно возникает вне зависимости от того, какой код (какой процесс) исполняется процессором в данный момент. Обработка аппаратного прерывания не должна учитывать, какой процесс является текущим.

1. Интерфейсы операционной системы: обработка исключительных ситуаций

**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 программные прерывания, являются синхронными событиями, возникающими в контексте текущей задачи.

Исключительные ситуации можно разделить на исправимые и неисправимые.

К **исправимым** относятся такие исключительные ситуации, как отсутствие нужной информации в оперативной памяти (нужной страницы памяти). После устранения причины исправимой исключительной ситуации (например, подкачки нужной страницы из swap'а) программа может продолжить выполнение. *Возникновение в процессе работы операционной системы исправимых исключительных ситуаций является нормальным явлением.*

**Неисправимые** исключительные ситуации обычно возникают в результате ошибок в программах. Например, деление на нуль или попытка доступа в адресное пространство другой программы. *Обычно «правильная» операционная система реагирует на такие ситуации завершением программы, вызвавшей неисправимую исключительную ситуацию.*

1. Интерфейсы операционной системы: работа с файлами

**Файл** - часть пространства на носителе информации, имеющая имя. Как правило, в этой именованной части пространства хранятся некоторые данные. Либо эта именованная часть пространства носителя резервируется для каких-либо целей.

Для хранения файлов на носителях создаются файловые системы.

**Файловая система** (file system) - метод организации хранения файлов на носителе информации, реализованный в некоторой операционной системе.

Почти всегда файловые системы разрабатываются в рамках проекта некоторой операционной системы и являются принадлежностью (частью) этой операционной системы.

Также существует зависимость между файловыми системами и носителями, на которых они могут использоваться.

1. Интерфейсы пользователя
2. Языки управления заданиями
3. Пакетная обработка
4. Командный интерпретатор в операционных системах UNIX
5. Командный интерпретатор в операционных системах Windows
6. Понятие процесса
7. Жизненный цикл процесса
8. Операции над процессами: одноразовые операции
9. Операции над процессами: многоразовые операции
10. Переключение контекста процессов
11. Потоки
12. Общие подходы к планированию процессов
13. Алгоритмы планирования процессов: First-Come, First-Served (FCFS)
14. Алгоритмы планирования процессов: Round Robin (RR)
15. Алгоритмы планирования процессов: Shortest-Job-First (SJF)
16. Алгоритмы планирования процессов: гарантированное планирование
17. Алгоритмы планирования процессов: приоритетное планирование
18. Алгоритмы планирования процессов: многоуровневые очереди
19. Алгоритмы планирования процессов: Многоуровневые очереди с обратной связью
20. Виды межпроцессного взаимодействия
21. Механизмы межпроцессного взаимодействия: прерывания
22. Механизмы межпроцессного взаимодействия: сообщения
23. Механизмы межпроцессного взаимодействия: именованные каналы
24. Механизмы межпроцессного взаимодействия: гнезда (сокеты)
25. Механизмы межпроцессного взаимодействия: общая память
26. Механизмы межпроцессного взаимодействия: семафоры
27. Межпроцессное взаимодействие в среде UNIX
28. Межпроцессное взаимодействие в Windows
29. Физическая организация памяти компьютера
30. Общие подходы к управлению памятью
31. Логическое и физическое адресное пространство
32. Порядок размещения процессов в памяти
33. Задача распределения памяти
34. Схемы организации памяти: виртуальная память
35. Схемы организации памяти: страничная организация памяти
36. Схемы организации памяти: сегментная и сегментно-страничная организация памяти
37. Свопинг
38. Ассоциативная память и иерархия памяти
39. Исключительные ситуации при работе с памятью
40. Стратегии управления страничной памятью
41. Задачи и классификация алгоритмов замещения страниц
42. Алгоритмы замещения страниц: алгоритм FIFO
43. Алгоритмы замещения страниц: оптимальный алгоритм
44. Алгоритмы замещения страниц: алгоритм LRU (выталкивание дольше всего не использовавшейся страницы)
45. Алгоритмы замещения страниц: алгоритм NFU (выталкивание редко используемой страницы)
46. Трешинг
47. Модель рабочего множества
48. Страничные демоны
49. Программная поддержка сегментной модели памяти процесса
50. Аспекты функционирования менеджера памяти
51. Общее понятие файловой системы
52. Файлы
53. Организация файлов и доступ к ним
54. Операции над файлами
55. Директории
56. Организация доступа к архиву файлов
57. Операции над директориями
58. Защита файлов
59. Общая структура файловой системы: структура системы хранения данных на дисках
60. Общая структура файловой системы: методы выделения дискового пространства
61. Общая структура файловой системы: управление свободным и занятым дисковым пространством
62. Общая структура файловой системы: размер логического блока, структура файловой системы на диске
63. Реализация директорий
64. Монтирование файловых систем
65. Современные архитектуры файловых систем
66. Связывание файлов
67. Кооперация процессов при работе с файлами
68. Hадёжность файловой системы
69. Производительность файловой системы