Вопросы к экзамену по дисциплине «Операционные системы» (5-й семестр, 2024 г.)

https://e-learning.bmstu.ru/iu5/pluginfile.php/3105/mod_folder/content/0/ОС_Вопросы%20к%20экзамену%20%282024%29.pdf?forcedownload=1

1. Операционная система (ОС)

Определение ОС:
Операционная система (ОС) – это комплекс программного обеспечения, который управляет аппаратными средствами компьютера, обеспечивает взаимодействие между пользователем, приложениями и оборудованием, а также предоставляет ресурсы для выполнения программ.

Место ОС в многоуровневой архитектуре вычислительной системы:
ОС находится между аппаратным уровнем и уровнем пользовательских приложений.

- **Нижний уровень** – оборудование (процессор, память, устройства ввода-вывода).
- **Средний уровень** – ОС, предоставляющая интерфейсы (API) и абстракции для работы с аппаратными средствами.
- **Верхний уровень** – приложения и программы пользователя.

Процессы ОС:
Процесс – это выполняющаяся программа, обладающая своим контекстом выполнения, включая код, данные, стек, регистры и другое.

- **Последовательный программный процесс** – процесс, выполняющий инструкции последовательно, одну за другой.
- **Формализованное описание процесса**: процесс можно представить как состояние системы, изменяемое набором операций (инструкций). Формализация включает:
    - P(t)=(s1,s2,...,sn)P(t) = (s_1, s_2, ..., s_n) – где sis_i – состояние на момент времени tt.

---

2. Отображение процессов ОС на вычислительные процессы

Отображение последовательного программного процесса:
ОС преобразует абстрактные задачи пользователя в операции на физическом оборудовании. Например, вызов системной функции записи на диск переводится в последовательность аппаратных команд.

Компиляция и интерпретация программ:

- **Компиляция**: перевод исходного кода программы в машинный код (выполняется до запуска).
- **Интерпретация**: выполнение программы построчно, анализируя исходный код в реальном времени.

---

3. Логическая и физическая модели программного процесса

Логическая модель:

- **Назначение**: Представляет процесс с точки зрения пользователя.
- **Состояния процесса**: готовность, выполнение, ожидание.

Граф состояний:
Готовность → Выполнение → Ожидание → Готовность

-

Физическая модель:

- **Назначение**: Описание того, как процессы отображаются на реальные ресурсы (процессор, память).
- **Состояния процесса**: загрузка контекста, переключение процессов, выполнение.

Граф состояния:
Загружен → Выполняется → Сохранен

-

---

4. Ресурсы ОС

Определение ресурса:
Ресурс ОС – это любой компонент системы, доступный для использования процессами (например, память, процессорное время, файлы).

Классификация ресурсов:

1. **По структуре**: простые (процессор), сложные (файловая система).
2. **По реальности существования**: реальные (оборудование), виртуальные (виртуальная память).
3. **По восстанавливаемости**: восстанавливаемые (память), невосстанавливаемые (время процессора).
4. **По характеру использования**: выделяемые (память), разделяемые (процессор).

---

5. Управление процессами ОС

Структуры данных:

- **Дескриптор процесса**: идентификатор, состояние, приоритет, ресурсы.
- **Контекст процесса**: регистры процессора, указатели на память, состояние стеков.
- **Таблица управления процессами**: массив дескрипторов процессов.

Операции над процессами:

- Создание, завершение, приостановка, возобновление.

---

6. Прерывания программных процессов

Назначение:
Прерывания обеспечивают реакцию на события (ошибки, запросы ввода-вывода, системные вызовы).

Типы прерываний:

1. **Внутренние**: ошибки деления на ноль, исключения.
2. **Внешние**: аппаратные сигналы (клавиатура, таймер).

Последовательность обработки:

1. Сохранение контекста.
2. Обработка прерывания (вызов обработчика).
3. Восстановление контекста.

Поддержка: аппаратная (регистры, контроллер прерываний) и программная (таблица векторов).

---

7. Синхронизация параллельных процессов ОС

Назначение синхронизации:
Обеспечение корректного доступа к общим ресурсам при параллельном выполнении процессов.

Типичные задачи синхронизации:

- **Взаимное исключение**: недопущение одновременного доступа.
- **Производители-потребители**: баланс между созданием и потреблением данных.
- **Читатели-писатели**: управление доступом на чтение и запись.
- **Круговое распределение ресурсов**: циклический доступ.

---

8. Аппаратная и программная реализация взаимоисключений

Аппаратная реализация:
Инструкции типа `Test-and-Set` и `Compare-and-Swap`.

Программная реализация:

- **Семафоры Дейкстры**: два типа операций:
  - `P()`: уменьшение значения семафора.
  - `V()`: увеличение значения семафора.
- **Программные каналы**: передача данных между процессами через очереди.

---

9. Дедлок (тупиковая ситуация)

Условия возникновения:

1. Взаимное исключение.
2. Удержание и ожидание.
3. Отсутствие принудительного изъятия.
4. Циклическое ожидание.

Стратегии предотвращения:

- Избегать захвата всех условий (например, заранее выделять ресурсы).
- **Иерархическое выделение**: ресурсы выделяются в фиксированном порядке.
- **Контролируемое выделение**: проверка возможности тупика перед выделением ресурса.

10. Планирование и выполнение программных процессов

Процесс и поток:
Процесс — это экземпляр выполняемой программы, включающий код, данные, стек и регистры процессора. Поток — это более легковесный компонент процесса, отвечающий за выполнение конкретной последовательности инструкций, но разделяющий с процессом ресурсы.

Мультипрограммные ОС:
В таких системах одновременно выполняется несколько процессов, что увеличивает загрузку процессора. Планировщик ОС определяет, какой процесс будет выполняться в данный момент.

Состояния процессов:

1. **Готовность** — процесс ожидает выделения процессора.
2. **Выполнение** — процесс получает процессорное время.

3. **Ожидание** — процесс ждет завершения операции ввода-вывода.

---

11. Отображение исходных модулей программ на оперативную память

Виртуальное адресное пространство (ВАП):
ВАП — это логическая область адресов, доступная процессу. Она позволяет изолировать процессы друг от друга.

Отображение ВАП на оперативную память:
Система преобразует виртуальные адреса в физические с помощью таблиц страниц или сегментов.

Смежное и несмежное размещение:

- **Смежное**: процесс размещается в одном непрерывном блоке памяти.
- **Несмежное**: память распределяется частями (например, страничная организация).

---

12. Управление оперативной и виртуальной памятью

Технология виртуальной памяти:
Виртуальная память позволяет процессам использовать больше памяти, чем физически доступно. ОС хранит данные на диске и загружает их в оперативную память по мере необходимости.

Страничная организация ВП:
ВАП разбивается на страницы, которые отображаются на блоки памяти (кадры). Это снижает фрагментацию памяти.

---

13. Диспетчеризация выполнения процессов

Однопроцессорные дисциплины:

- **Невытесняющее планирование**: процесс выполняется до завершения.
- **Вытесняющее планирование**: процесс может быть прерван.

Приоритетные дисциплины:
Процессы ранжируются по приоритетам, и более приоритетный процесс получает ресурсы.

Многопроцессорные дисциплины:
Процессы распределяются между несколькими процессорами для повышения производительности.

---

14. Управление вводом-выводом

Типы устройств:

- Устройства ввода (клавиатура, сканеры).
- Устройства вывода (мониторы, принтеры).
- Комбинированные устройства (жесткие диски).

Контроллеры устройств:
Аппаратные модули, обеспечивающие взаимодействие между устройствами и системой.

---

15. Технология ввода-вывода

Методы передачи данных:

1. **Программируемый ввод-вывод**: процессор управляет всеми операциями.
2. **Ввод-вывод с прерываниями**: обработка данных начинается после прерывания от устройства.
3. **Прямой доступ к памяти (DMA)**: передача данных между устройством и памятью без участия процессора.

---

16. Программное обеспечение ввода-вывода

Архитектура подсистемы:

- Подсистема управления вводом-выводом состоит из драйверов, системных таблиц и API.
- Драйверы обеспечивают взаимодействие ОС с аппаратными устройствами.

---

17. Логическая организация файлов

Определение файла:
Файл — именованная область данных на диске, организованная для долговременного хранения информации.

Каталоги:
Иерархическая структура, позволяющая упорядочивать файлы.

Атрибуты файла:
Имя, размер, права доступа, дата создания.

---

18. Физическая реализация файловой системы

Файловая система:
Метод организации данных на носителях.

- **Раздел**: физическая часть диска.
- **Логический том**: логическое представление раздела.

Форматирование:
Подготовка раздела для хранения данных.

---

19. Хранение атрибутов и данных файлов

Каталоги и индексные узлы:
Каталоги хранят информацию о файлах, а индексные узлы (i-node) — метаданные файла.

Адресация данных:
Данные файла адресуются по прямым, косвенным или многокосвенным ссылкам.

---

20. Архитектуры операционных систем

Монолитные ОС:
Все функции реализуются в одном блоке кода, что обеспечивает высокую производительность, но усложняет модификацию.

Многоуровневые ОС:
Система разделена на уровни (аппаратный, пользовательский), что улучшает структуру и безопасность.

Микроядерные ОС:
Основные функции ОС минимизированы (управление памятью, процессы), а другие реализованы как модули.

21. ОС Unix: особенности, архитектура и модули ядра

Особенности ОС Unix:
Unix отличается высокой степенью модульности, переносимости и многозадачности. Эта ОС широко используется для серверов, рабочих станций и встраиваемых систем. Основные характеристики:

- Многопользовательский доступ.
- Многозадачность.
- Иерархическая файловая система.
- Простота и мощь текстовых интерфейсов.
- Использование открытых стандартов.

Архитектура ОС Unix:
Unix имеет трехуровневую архитектуру:

1. **Аппаратное обеспечение** — устройства, доступ к которым обеспечивается через ядро.
2. **Ядро (Kernel)** — центральная часть системы, управляющая процессами, памятью, файловыми системами и вводом-выводом.
3. **Утилиты и оболочки** — инструменты для взаимодействия пользователя с системой.

Основные модули ядра:

- **Управление процессами**: создание, завершение, планирование и синхронизация процессов.
- **Управление памятью**: распределение и освобождение памяти, поддержка виртуальной памяти.
- **Файловая система**: обеспечение доступа к файловым структурам и устройствам хранения.
- **Подсистема ввода-вывода**: управление драйверами и аппаратными устройствами.
- **Сетевая подсистема**: реализация сетевых протоколов.

---

22. ОС Unix: режимы работы и многопользовательская поддержка

Режимы работы системы:

- **Пользовательский режим**: ограниченный доступ к ресурсам системы, выполняются пользовательские приложения.
- **Режим ядра**: полный доступ ко всем ресурсам системы, выполняются функции ядра.

Многопользовательский режим:
Unix позволяет нескольким пользователям работать на одной системе одновременно. Каждый пользователь имеет уникальный идентификатор (UID) и может выполнять свои процессы независимо от других.

Использование физических, виртуальных и псевдотерминалов:

- **Физические терминалы**: устройства, непосредственно подключенные к системе (например, клавиатура, монитор).
- **Виртуальные терминалы**: логические интерфейсы, создаваемые внутри системы для работы нескольких пользователей.
- **Псевдотерминалы (PTY)**: создаются для обеспечения связи между процессами (например, при удаленном доступе через SSH).

---

23. ОС Unix: управление процессами

Структуры данных для управления процессами:

- **Дескриптор процесса**: хранит UID, PID, состояние, приоритет, счетчик программ, указатели на ресурсы.
- **Таблицы процессов**: глобальные структуры, содержащие информацию обо всех процессах.

Уровни выполнения процессов:

- **Пользовательский уровень**: выполняются прикладные программы.
- **Уровень ядра**: обработка системных вызовов, выполнение функций ядра.

Граф состояний процесса:

1. **Создан**: процесс ожидает инициализации.
2. **Готов**: процесс готов к выполнению.
3. **Выполняется**: процесс получает процессорное время.
4. **Ожидает**: процесс ждет завершения операции ввода-вывода.
5. **Завершен**: процесс освобождает ресурсы и удаляется из таблицы.

---

24. ОС Unix: управление вводом-выводом

Компоненты ввода-вывода:

- **Драйверы устройств**: программы, обеспечивающие взаимодействие ядра с аппаратными устройствами.
- **Буферизация**: временное хранение данных между устройством и процессом.
- **Таблицы ввода-вывода**: структуры, содержащие информацию о текущих операциях.

Типы драйверов:

- Блочные (жесткие диски).
- Символьные (клавиатура, принтеры).
- Сетевые (карты Ethernet).

Подсистема STREAMS:
STREAMS позволяет создавать модульные драйверы, обеспечивающие двунаправленный ввод-вывод.

---

25. ОС Linux: архитектура и подсистемы ядра

Архитектура системы:
Linux имеет модульную архитектуру, схожую с Unix, но позволяет гибко добавлять или удалять модули ядра.

Основные подсистемы ядра:

- **Планировщик процессов**.
- **Виртуальная память**.
- **Файловая система VFS**.
- **Сетевая подсистема**.
- **Подсистема драйверов устройств**.

Уровни выполнения ОС:

- **Пользовательский уровень**: запуск прикладных программ.
- **Уровень ядра**: управление аппаратными ресурсами.

Многопользовательский режим:
Linux поддерживает одновременный доступ нескольких пользователей с разграничением прав.

---

26. ОС Linux: типы пользователей и управление

Типы пользователей:

- **Обычные пользователи**: ограниченные права доступа.
- **Системные пользователи**: выполняют системные функции.
- **Суперпользователь (root)**: имеет полный доступ к системе.

Управление пользователями:

- Добавление, удаление, изменение учетных записей.
- Управление группами пользователей.

Права суперпользователя:
Root может выполнять задачи администрирования, такие как управление процессами, настройка сетей, изменение системных параметров.

---

27. ОС Linux: процессы и потоки

Понятие процесса и потока:
Процесс — это экземпляр программы в выполнении. Поток — это легковесная часть процесса, которая может выполняться параллельно.

Создание процессов и потоков:

- **fork()**: создает новый процесс.
- **pthread_create()**: создает поток.

Классы процессов:

- Реальные (пользовательские).
- Системные (демоны).

Виртуальное адресное пространство:
Процесс получает логическое адресное пространство, которое отображается на физическую память.

---

28. ОС Linux: уровни выполнения процессов

- **Уровень пользователя**: выполняются программы.
- **Уровень ядра**: выполняются системные вызовы и операции с ресурсами.

---

29. ОС Linux: управление виртуальной памятью

Страничное распределение памяти:
Физическая память делится на блоки (кадры), которые соответствуют страницам виртуальной памяти.

Разделы и файлы подкачки:
Используются для временного хранения данных, не помещающихся в оперативную память. Это позволяет увеличить объем доступной памяти.

---

30. ОС Linux: управление процессами

Таблица управления процессами:
Содержит информацию о каждом процессе (PID, состояние, ресурсы).

Состояния процесса:

- Создан.
- Готов.

- Выполняется.
- Ожидание.
- Завершен.

Граф состояний:
Определяет переходы между состояниями в зависимости от событий.

---

31. ОС Linux: планирование выполнения процессов

Планировщик процессов:
Ключевая часть ядра, распределяющая процессорное время между процессами.

Политики планирования:

- **CFS (Completely Fair Scheduler)**: равномерное распределение ресурсов.
- **RT (Real-Time)**: для процессов реального времени.

Приоритеты процессов:
Процессы могут иметь базовый и динамический приоритет, определяющий порядок их выполнения.

32. ОС Linux: Логическая организация файловых систем

Логическая организация файловых систем:
Файловая система в Linux представляет собой способ хранения и организации данных на носителях информации. Логическая организация включает иерархическую структуру каталогов, в которой файлы и каталоги расположены по дереву, начиная с корневого каталога `/`.

Типы файлов:

1. **Обычные файлы**: текстовые, бинарные, исполняемые файлы.
2. **Каталоги**: специальные файлы, содержащие записи о файлах и подкаталогах.
3. **Символьные ссылки**: указывают на другой файл или каталог.
4. **Жесткие ссылки**: создают несколько имен для одного и того же файла.
5. **Специальные файлы устройств**:
    - **Блочные устройства**: доступ к данным блоками (например, диски).
    - **Символьные устройства**: посимвольный доступ (например, клавиатура).
6. **Сокеты**: файлы для межпроцессного взаимодействия.
7. **Файлы FIFO (именованные каналы)**: используются для передачи данных между процессами.

Монтирование файловых систем:
Монтирование связывает файловую систему с конкретной директорией.

- Команда: `mount` — для подключения.

- Команда: `umount` — для отключения.
  Монтирование может быть автоматическим (при загрузке системы через `/etc/fstab`) или ручным.

---

33. ОС Linux: Физическая реализация файловых систем

Структура файловой системы:
Файловая система состоит из нескольких ключевых элементов:

- **Суперблок**: содержит информацию о самой файловой системе (размеры, состояние, параметры).
- **Блоки данных**: хранят содержимое файлов.
- **Индексные узлы (inode)**: содержат метаданные файла (размер, права доступа, указатели на блоки данных).
- **Каталоги**: таблицы, связывающие имена файлов с их индексными узлами.

Хранение атрибутов и данных:

- **Атрибуты**: права доступа, время модификации, владелец и группа — хранятся в `inode`.
- **Данные файлов**: расположены в блоках данных, ссылки на которые находятся в `inode`.

---

34. Планирование пространства дисковой подсистемы

Создание дисковых разделов:
Диск разделяется на логические области (разделы), каждая из которых может содержать свою файловую систему.

- Типы разделов: первичные, расширенные и логические.

Технологии управления разделами:

- **PT (Partition Table)**: устаревшая схема разметки, поддерживает до 4 первичных разделов.
- **GPT (GUID Partition Table)**: современная схема, поддерживает до 128 разделов, увеличенные размеры томов.

Форматирование и монтирование файловых систем:

1. Форматирование: создается файловая система (`mkfs.ext4`, `mkfs.xfs` и др.).
2. Монтирование: связывание с директорией (`mount`).
3. Запись в `/etc/fstab`: автоматизация монтирования.

---

35. Управление дисковым пространством с LVM

LVM (Logical Volume Manager):
Технология управления дисковым пространством, позволяющая гибко изменять размеры томов.

Основные компоненты LVM:

- **Физические тома (PV)**: физические устройства или разделы.
- **Группы томов (VG)**: объединение физических томов.
- **Логические тома (LV)**: создаются внутри группы томов и используются для файловых систем.

Создание файловых систем:

1. Создать физический том: `pvcreate /dev/sdX`.
2. Создать группу томов: `vgcreate VG_NAME /dev/sdX`.
3. Создать логический том: `lvcreate -L SIZE -n LV_NAME VG_NAME`.
4. Создать файловую систему: `mkfs.ext4 /dev/VG_NAME/LV_NAME`.

Изменение размеров:

- Увеличение: `lvextend` и `resize2fs`.
- Уменьшение: `lvreduce` и `resize2fs`.

---

36. Особенности файловых систем ext2, ext3, ext4

1. **ext2**:
    - Без журналирования.
    - Высокая производительность, но медленное восстановление после сбоев.
2. **ext3**:
    - Добавлено журналирование.
    - Быстрое восстановление после сбоев.
    - Обратная совместимость с ext2.
3. **ext4**:
    - Поддержка больших объемов данных.
    - Увеличенные размеры файлов и томов.
    - Эффективное использование дискового пространства (разреженные файлы).

---

37. Политика безопасности файловых систем ext

Контроль доступа:
Файловые системы ext используют модель POSIX для управления правами доступа.

Файловые разрешения:
Каждый файл имеет три набора разрешений: для владельца, группы и остальных.

- Чтение (r), запись (w), выполнение (x).

Атрибуты файлов:
Дополнительные параметры, управляющие поведением файлов (`lsattr`, `chattr`).

Администрирование прав:

- Команды: `chmod`, `chown`, `chgrp`.

---

38. Расширенные права доступа (setuid, setgid, sticky bit)

setuid: выполнение программы с правами владельца.
setgid: выполнение программы с правами группы.
sticky bit: предотвращение удаления файлов в общем каталоге, кроме владельцем.

Установка битов:

Команда `chmod`:
```
chmod u+s file   # setuid
chmod g+s file   # setgid
chmod +t dir     # sticky bit
```

-

---

39. Использование ACL для управления доступом

Access Control List (ACL):
Позволяет задавать права для отдельных пользователей или групп, расширяя стандартные разрешения.

Управление ACL:

- Просмотр: `getfacl`.
- Установка: `setfacl`.
- Удаление: `setfacl -x`.

Пример:

setfacl -m u:username:rw file
getfacl file

---

40. Совместное использование файлов (жесткие и символические ссылки)

Жесткие ссылки (hard link):

- Указывают на тот же индексный узел (`inode`), что и оригинальный файл.
- Файл и ссылка равноправны.
- Не работают для каталогов и файлов на разных файловых системах.

Символические ссылки (soft link):

- Указывают на путь к файлу.
- Работают с каталогами и файлами на разных файловых системах.
- Если оригинальный файл удален, ссылка становится "битой".

Достоинства и недостатки:

- Жесткие ссылки: быстрые, но ограничены одной файловой системой.
- Символические ссылки: универсальны, но более медлительны и подвержены ошибкам при удалении целевого файла.

Вопросы к экзамену по дисциплине «Операционные системы» (ВО, 2024 г.)

https://e-learning.bmstu.ru/iu5/pluginfile.php/16059/mod_folder/content/0/OC_BO_Вопросы%20к%20экзамену%20%282024%29.pdf?forcedownload=1

1. Определение операционной системы (ОС). Место ОС в многоуровневой архитектуре вычислительной системы. Процессы ОС. Последовательный программный процесс. Формализованное описание программного процесса.

Операционная система (ОС) — это комплекс программ, управляющий аппаратными средствами компьютера и предоставляющий платформу для выполнения прикладных программ. ОС обеспечивает управление ресурсами (процессор, память, устройства ввода-вывода) и организует взаимодействие пользователя с компьютером.

Место ОС в архитектуре: ОС располагается между аппаратным обеспечением и прикладными программами, обеспечивая многоуровневую архитектуру:

- **Уровень 0**: Аппаратное обеспечение.
- **Уровень 1**: Операционная система.

- **Уровень 2**: Системные библиотеки и сервисы.
- **Уровень 3**: Прикладные программы.

Процессы ОС:
Процесс — это программа в состоянии выполнения, обладающая своим адресным пространством, состоянием и ресурсами. ОС управляет процессами с использованием таблиц процессов, дескрипторов и других структур данных.

Последовательный программный процесс:
Процесс, выполняющийся в строгом порядке, определяемом программой. Прерывания отсутствуют, что обеспечивает детерминированное выполнение.

Формализованное описание программного процесса:
Формализованная модель включает:

- **Состояние процесса**: набор текущих данных и контекст выполнения.
- **Граф состояний**: модель переходов между состояниями (запущен, готов, ожидает).

---

2. Отображение программного процесса ОС на вычислительный процесс вычислительной системы. Компиляция и интерпретация программ.

Отображение программного процесса на вычислительный:
ОС выделяет ресурсы (ЦП, память, устройства ввода-вывода) для выполнения программного процесса. Программный процесс преобразуется в вычислительный за счет выполнения его инструкций процессором.

Компиляция:
Компиляция — преобразование исходного кода программы в машинный код перед выполнением. Пример: компиляторы `gcc`, `clang`.

Интерпретация:
Интерпретация — построчное выполнение кода без предварительного перевода в машинный код. Пример: интерпретаторы Python, JavaScript.

---

3. Логическая и физическая модели программного процесса

Логическая модель:

- **Назначение**: абстрактное представление процесса с точки зрения ОС.
- **Состояния процесса**: готов, выполняется, ожидает.
- **Граф состояний**:
    - **Готов → Выполняется**: процесс получает процессор.
    - **Выполняется → Ожидает**: процесс ждет ресурс.

    - **Ожидает → Готов**: ресурс освобожден.

Физическая модель:

- **Назначение**: отображение процесса на конкретные аппаратные ресурсы.
- **Состояния**: включает конкретные аппаратные аспекты, например, управление контекстом и регистровым состоянием.
- **Граф состояний**: схож с логической моделью, но учитывает привязку к физическим ресурсам.

---

4. Понятие ресурса ОС

Ресурс ОС: объект, управляемый ОС и используемый процессами. Примеры: процессорное время, память, устройства ввода-вывода.

Параметры ресурса:

- Доступность.
- Емкость.
- Скорость.

Классификация ресурсов:

1. **По структуре**:
    - Простые: процессорное время.
    - Составные: устройства ввода-вывода.
2. **По реальности существования**:
    - Реальные: оперативная память.
    - Логические: файлы.
3. **По восстанавливаемости**:
    - Восстанавливаемые: процессорное время.
    - Невосстанавливаемые: энергоемкие устройства.
4. **По характеру использования**:
    - Эксклюзивные: используются одним процессом.
    - Разделяемые: используются несколькими процессами.

---

5. Прерывания программных процессов

Назначение механизма прерываний:
Прерывания позволяют процессору реагировать на события, изменяя последовательность выполнения инструкций.

Типы прерываний:

- **Внутренние**: деление на ноль, ошибки памяти.

- **Внешние**: сигналы от устройств ввода-вывода.

Последовательность обработки прерываний:

1. Сохранение текущего контекста.
2. Переход к обработчику.
3. Выполнение обработчика.
4. Восстановление контекста.

Аппаратная и программная поддержка:

- **Аппаратная**: контроллеры прерываний.
- **Программная**: драйверы устройств, обработчики прерываний.

---

6. Синхронизация параллельных процессов ОС

Назначение синхронизации:
Обеспечивает корректное взаимодействие процессов, предотвращая конфликты при доступе к разделяемым ресурсам.

Типичные задачи синхронизации:

1. **Взаимное исключение**: одновременный доступ исключен.
2. **Производители-потребители**: сбалансированный обмен данными.
3. **Читатели-писатели**: разрешение конфликтов при доступе.
4. **Круговое распределение ресурсов**: предотвращение дедлоков.

---

7. Программная реализация синхронизации

Взаимоисключения:

- Использование блокировок (mutex).

Семафоры Дейкстры:

- Два типа: счетные и двоичные.
- Примитивы: `wait`, `signal`.

Программные каналы:

- Позволяют процессам обмениваться сообщениями через очередь сообщений.

---

8. Планирование и выполнение процессов

Процесс: программа в состоянии выполнения.
Поток: легковесный процесс, часть процесса, использующая общую память.

Мультипрограммные ОС: позволяют выполнять несколько процессов одновременно за счет разделения времени процессора.

Состояния процессов:

1. Готов.
2. Выполняется.
3. Ожидает.

---

9. Виртуальное адресное пространство и виртуальная память

Виртуальное адресное пространство (ВАП):
Абстракция памяти, предоставляющая каждому процессу изолированное пространство.

Технология виртуальной памяти (ВП):
Позволяет использовать больше памяти, чем физически доступно, за счет использования диска.

Страничная организация:

- Память делится на страницы.
- Страницы отображаются на фреймы физической памяти.
- Таблицы страниц управляют отображением.

---

10. Диспетчеризация выполнения процессов

Диспетчеризация: процесс выбора следующего для выполнения процесса.

Однопроцессорные дисциплины:

- Невытесняющее: процесс завершает работу до переключения.
- Вытесняющее: процесс может быть приостановлен.

Приоритетные дисциплины:

- Высокий приоритет завершает первым.

Многопроцессорные дисциплины:

- Балансировка нагрузки между процессорами.

11. Архитектура внешних устройств ввода-вывода

Типы устройств ввода-вывода:

1. **Блочные устройства**: диски, SSD, работающие с блоками фиксированного размера.
2. **Символьные устройства**: клавиатуры, мыши, последовательные порты, работающие с символами или байтами.
3. **Сетевые устройства**: сетевые интерфейсы для передачи данных между системами.

Методы передачи данных:

- **Программируемый ввод-вывод (PIO)**: процессор напрямую управляет передачей.
- **Ввод-вывод, управляемый прерыванием**: процессор прерывается устройством после завершения передачи данных.
- **Прямой доступ к памяти (DMA)**: данные передаются между устройством и памятью без участия процессора.

Аппаратные компоненты ввода-вывода:

- **Контроллеры устройств**: интерфейс между ОС и устройством, обрабатывает команды ОС.
- **Порты ввода-вывода**: каналы для взаимодействия ОС с устройствами.

Настройка контроллеров: включает настройку параметров, таких как режимы передачи, скорости, прерывания.

Способы доступа к портам контроллера:

1. **Прямой доступ (I/O Ports)**: работа через аппаратные порты.
2. **Память с отображением I/O (Memory-Mapped I/O)**: управление через адресное пространство памяти.

---

12. Технология ввода-вывода

Обмен данных:

1. **Программируемый ввод-вывод (PIO)**: ЦП выполняет все операции передачи данных, работает медленно.
2. **Ввод-вывод, управляемый прерыванием**: устройство прерывает процессор, сигнализируя о завершении передачи. Это снижает нагрузку на процессор.
3. **Прямой доступ к памяти (DMA)**: устройство напрямую передает данные в память, освобождая ЦП для других задач.

---

13. Логическая организация файлов и файловых систем

Определение файла: именованная область хранения данных.

Атрибуты файла:

- Имя.
- Размер.
- Дата и время создания/изменения.
- Права доступа.

Каталог: структурированное хранилище, содержащее ссылки на файлы и другие каталоги.

Логическая организация данных файла:

1. Последовательный доступ.
2. Директ-доступ.
3. Индексированный доступ.

Интерфейс файловой системы: набор системных вызовов для работы с файлами и каталогами (открытие, чтение, запись, закрытие).

---

14. Физическая реализация файловой системы

Раздел: физическая или логическая часть диска, содержащая файловую систему.

Логический том: объединяет несколько разделов или устройств в единую файловую систему.

Форматирование логических томов: процесс разметки тома для хранения данных (создание метаданных, структуры каталогов, блоков).

---

15. Физическая реализация хранения атрибутов и данных файлов

Каталоги: хранят ссылки на файлы и метаданные.

Индексные узлы (inodes): структуры данных, хранящие атрибуты и расположение файлов.

Способы адресации блоков данных:

1. Прямой (direct addressing).
2. Косвенный (single/double indirect).
3. Комбинированный (multilevel indexing).

---

16. Установка гостевой ОС Альт Рабочая станция в VirtualBox

Этапы установки:

1. Создание виртуальной машины (определение параметров CPU, памяти, дисков).
2. Настройка виртуальных устройств ввода-вывода.
3. Установка с образа ISO.

Уровни выполнения ОС:

1. **Пользовательский**: выполнение приложений.
2. **Ядро**: управление ресурсами и аппаратурой.

Виртуальные консоли: текстовые интерфейсы управления системой, доступные через комбинации клавиш (`Ctrl + Alt + F1-F6`).

---

17. Типы пользователей Alt Linux

Типы пользователей:

1. **Обычные**: ограниченные права доступа.
2. **Суперпользователь (root)**: полный доступ к системе.

Создание учетных записей:
Используются команды `adduser`, `useradd`.

Получение прав root:

1. Вход в систему как root.
2. Использование команды `sudo`.

Настройка sudo: изменение файла `/etc/sudoers` для ограничения прав доступа.

---

18. Планирование пространства дисковой подсистемы в Alt Linux

Создание разделов: инструменты `fdisk`, `parted`.

Технологии управления:

- PT (Partition Table): старая схема разметки.
- GPT (GUID Partition Table): современная схема, поддерживает большие объемы.

Форматирование и монтирование:

1. Форматирование: команда `mkfs`.
2. Монтирование: команда `mount`.

Виртуальные файловые системы: абстракция, упрощенная структура (например, `/proc`, `/sys`).

---

19. Управление дисковым пространством с LVM

LVM (Logical Volume Manager): динамическое управление дисковым пространством.

Компоненты:

1. Физические тома (Physical Volumes, PV).
2. Группы томов (Volume Groups, VG).
3. Логические тома (Logical Volumes, LV).

Операции:

- Создание: команды `pvcreate`, `vgcreate`, `lvcreate`.
- Увеличение/уменьшение размеров: команды `lvextend`, `lvreduce`.

---

20. Политика безопасности файловых систем ext

Контроль доступа:

- **Разрешения (permissions)**: чтение (r), запись (w), выполнение (x).
- **Атрибуты (attributes)**: настройки поведения файлов.

Файловые разрешения:

- Уровни: владелец, группа, остальные.
- Настройка: команда `chmod`.

Расширенные разрешения:

- **SetUID**: выполнение от имени владельца.
- **SetGID**: выполнение от имени группы.
- **Sticky bit**: защита файлов в общедоступных каталогах.

Настройка: команды `chown`, `chgrp`, `setfacl`.

21. Политика безопасности файловых систем ext. Расширенные права доступа к файлам и каталогам ОС Alt Linux

Биты расширенных разрешений:

1. **SetUID (Set User ID)**:
    - При выполнении программы от имени обычного пользователя, процесс получает права владельца файла.
    - Применяется к исполняемым файлам.
    - Установка: `chmod u+s <файл>`.
2. **SetGID (Set Group ID)**:
    - Программы или каталоги наследуют группу владельца файла или каталога, а не текущую группу пользователя.
    - Используется для управления доступом к файлам в группах.
    - Установка: `chmod g+s <файл/каталог>`.
3. **Sticky bit**:
    - Применяется к каталогам. Позволяет пользователям создавать файлы, но запрещает их удаление другими пользователями (кроме владельца или root).
    - Пример: каталог `/tmp`.
    - Установка: `chmod +t <каталог>`.

Установка битов:
Биты задаются с помощью команды `chmod` с символическим (`+s`, `+t`) или числовым методом.
Пример числового: `chmod 1755 <файл>` (1 — sticky bit, 4 — setuid, 2 — setgid).

---

22. Политика безопасности файловых систем ext. Назначение прав по стандарту POSIX и ACL

Стандарт POSIX:
POSIX определяет права доступа через три группы:

- Владелец (user).
- Группа (group).
- Остальные (others).

Списки контроля доступа (ACL):
Расширяют стандартную модель POSIX, позволяя задавать права для отдельных пользователей и групп.

Основные команды ACL:

1. **Установка ACL**:
    - Команда: `setfacl`.

    - Пример: `setfacl -m u:username:rwx <файл>` (добавление прав для конкретного пользователя).
2. **Просмотр ACL**:
    - Команда: `getfacl`.
    - Пример: `getfacl <файл>`.
3. **Удаление ACL**:
    - Команда: `setfacl -x u:username <файл>`.

Применение ACL:

- Используется для управления сложными правами доступа.
- Пример: права для нескольких пользователей, которые не принадлежат одной группе.

---

23. ОС Alt Linux. Совместное использование каталогов и файлов

Жесткие ссылки (Hard Links):

- Ссылка указывает на один и тот же индексный узел (inode) файла.
- Используются для обеспечения доступа к одному и тому же содержимому файла.
- Удаление оригинального файла не влияет на жесткую ссылку.

Создание: `ln <оригинальный_файл> <ссылка>`.

Ограничения:

- Нельзя создавать ссылки на каталоги (по умолчанию).
- Работают только в пределах одного раздела.

Преимущества:

- Экономия дискового пространства.
- Защита от случайного удаления данных (пока есть ссылки).

Недостатки:

- Отсутствие гибкости (нельзя ссылаться между разделами).

Символические ссылки (Soft Links):

- Указывают на путь к файлу, а не на inode.
- Если оригинальный файл удален, символическая ссылка становится неработоспособной (мёртвой).

Создание: `ln -s <оригинальный_файл> <ссылка>`.

Преимущества:

- Можно ссылаться на файлы между разделами.
- Возможность ссылок на каталоги.

Недостатки:

- Занимают дополнительное пространство для хранения пути.
- Зависимость от пути файла.

Применение в ОС Alt Linux:

- **Жесткие ссылки** используются для дублирования файлов с сохранением целостности данных.
- **Символические ссылки** удобны для работы с файлами и каталогами, находящимися в разных местах файловой системы.