Билеты к экзамену (от Сакулина И.М. К3121)

Теоретические:

1) Понятие операционной системы (ОС). Задачи ОС.

Операционная система — программное обеспечение, управляющее аппаратным обеспечением компьютера и позволяющее запускать на них прикладные программы. Совокупность ядра операционной системы и работающих поверх него программ и утилит.

Задачи ОС

- Реализация нескольких уровней абстракции для пользователя и пользовательских программ
  - Абстрагирование сложностей низкоуровневого программирования
- Взаимодействие с оборудованием (Процессор, Прерывания, I/O, внешними устройствами)
- Управление ресурсами (процессами, памятью, файловыми системами)
  - Распределение и управление процессами
  - Организация доступа к памяти
  - Управление файловыми системами
  - Выполнение и завершение процессов
- Защита данных и безопасность
  - Шифрование и контроль доступа
  - Защита от вредоносных программ
  - Защита от сбоев и восстановление
- Сетевое взаимодействие
  - Управление соединениями, поддержка сетевых протоколов
  - Маршрутизация, формирование и доставка сетевых пакетов
  - Безопасность сетевого взаимодействия
- API
  - Набор правил для доступа к системным ресурсам через ОС
  - Позволяет запрашивать услуги ОС без знания деталей реализации
- Предоставление пользовательского интерфейса
  - GUI (графический интерфейс), командный интерфейс и другие

2) Linux как ОС (тип архитектуры), особенности лицензирования, ядро и дистрибутив.

Linux как ОС – *UNIX-подобная ОС* с открытым исходным кодом, представляющая собой архитектуру *монолитного ядра*.

Особенности *UNIX-подобных систем*

- Используют текстовые файлы для конфигурации
- Широко применяют командную строку
- Используют конвейеры для обработки данных
- Ядро представляет собой файловую структуру
- Используют язык C для системного программирования
- Поддерживают мультиплатформенность

Linux распространяется по **свободной лицензии GNU General Public License (GPL).**

Основные принципы лицензии GPL: *авторское право, бесплатность, открытость*.

Основные права пользователей: использовать без ограничений, копировать и распространять, модифицировать исходный код.

Обязанности разработчиков:

- должны предоставлять полный исходный код при распространении,
- не могут устанавливать дополнительных ограничений на использование или распространение,
- должны указывать копирайт и условия лицензии.

Linux –это ядро!

Монолитное ядро, но с загружаемыми модулями! Разделяемые системные библиотеки (system libraries) содержат стандартный набор функций, используемых приложениями для запросов к системным сервисам ядра.

- Управление: Линус Торвальдс
- Ответственные за версию ядра – Мейнтейнеры, модулей – Сообщество
- Тоталитарная демократия
- Любые изменения проходят много уровней тестирования. Сообщество активно участвует в этом процессе, выявляя и исправляя ошибки.
- Ядро Linux выпускается в виде стабильных версий каждые несколько месяцев. Между основными версиями существуют тестовые (релиз-кандидаты), которые помогают выявить и устранить ошибки до выхода финальной версии.
- Система контроля версий - Git.
- Финансирование Linux Faundation

Дистрибутив Linux — это сборка, включающая ядро Linux, набор системных утилит и программ, а также пользовательский интерфейс.

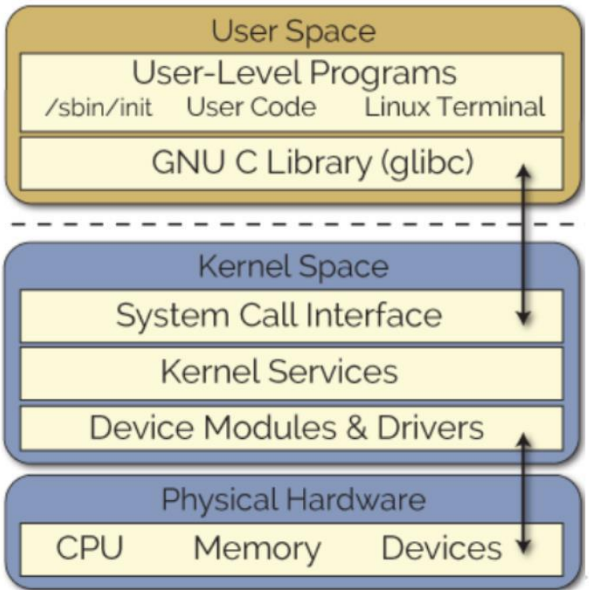
Модели распространения дистрибутов Linux

- *Выпуски с фиксированными релизами (Fixed Release)*, новые версии через определенные промежутки времени (Ubuntu).
- *Непрерывные релизы (Rolling Release)*. Обновления происходят постоянно. Пользователи получают новейшие версии программного обеспечения (Arch Linux, CentOS 9, Fedora)
- *Гибридные релизы (Hybrid Release)*. Выпускаются регулярно обновляемые пакеты в пределах стабильного релиза (Debian с тремя ветками Stable, Testing и Unstable).
- *Дистрибутивы с поддержкой корпоративного уровня (Enterprise Release)*. Принцип Fixed Release, но с длительными периодами поддержки (Red Hat Enterprise Linux (RHEL) и SUSE Linux Enterprise Server (SLES).

3) Обобщенная архитектура Linux

Монолитное ядро, но с загружаемыми модулями!

- **Process Scheduler** - планировщик процессов (делит CPU),
- **Memory Manager** - система управления памятью - формирует из оперативной памяти и свопа виртуальную память, делит ее на kernel и user space,
- **Inter-Process Communication** - механизмы межпроцессного взаимодействия,
- **Virtual File System** - виртуальная файловая система,
- **Network Interface** - сетевая подсистема - реализует концепции сетевых интерфейсов и стеков сетевых протоколов

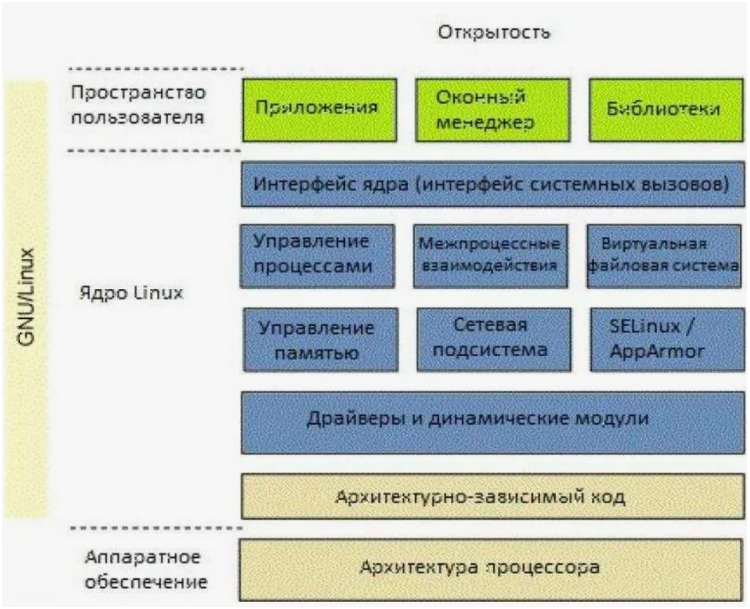


Linux как ОС – *UNIX-подобная ОС*, представляющая собой *монолитное ядро*.

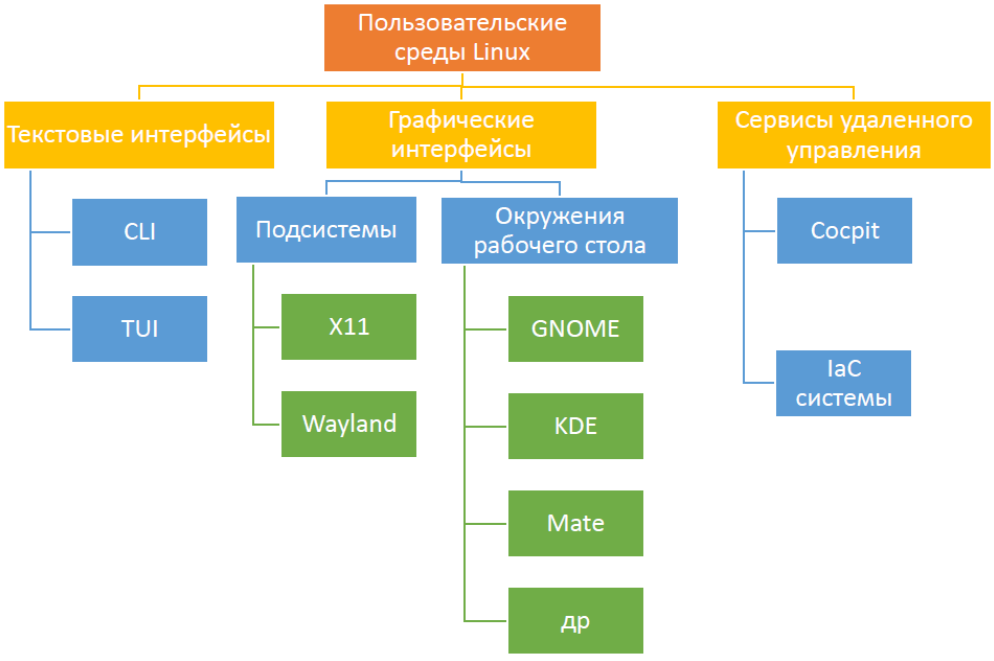
Программы user-space - библиотеки, утилиты, прикладное ПО, службы (сервисы, демоны).

Glibc предоставляет реализацию многих стандартных функций языка C, описанных в стандартах ISO C и POSIX. Это включает в себя обработку ввода-вывода, работу с файлами, управление памятью, математические операции, строковые функции и многое другое. Обеспечивает переносимость.

Системные вызовы - выполняются в ядре - функции операционной системы, реализуемые ядерными компонентами и доступные внеядерным компонентам



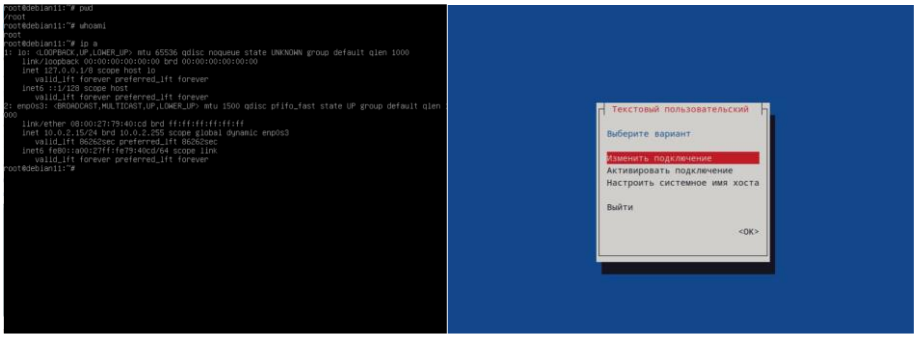
4) Типы UI в Linux (текстовой интерфейс, TUI и GUI, примеры графических оболочек, X-server vs Wayland)



Текстовый интерфейс

CLI (Command Line Interface) – интерфейс командной строки, использующий текстовые команды для ввода и вывода информации.

TUI (Text User Interface) – более интерактивный текстовый интерфейс, имитирующий графические элементы через символы.

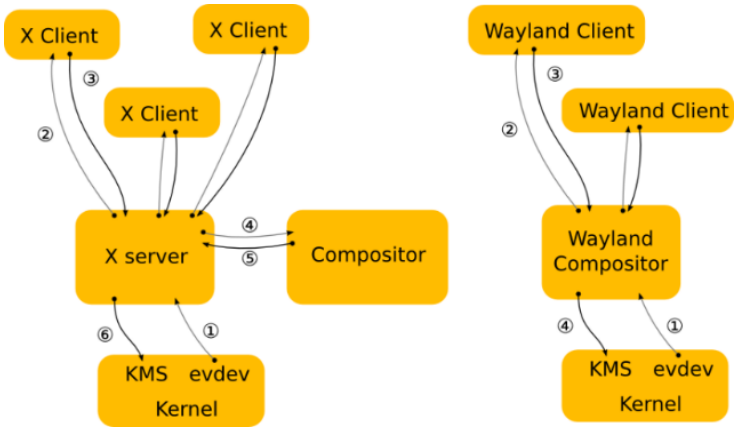


Графический интерфейс GUI (Graphical User Interface) – визуальная оболочка, позволяющая пользователю взаимодействовать с компьютером через графические элементы.

Подсистемы предоставляют базовые функции отображения (основа графического окружения).

X-server

– Стандартная технология отображения с 1987 года
– Использует клиент-серверный модель с взаимодействием приложений через X сервер
– Более сложная архитектура с несколькими компонентами
– Лучшая сетевая прозрачность - может отображать окна из удаленных машин
– Широко поддерживается графическими драйверами, особенно в играх
– X-server все еще имеет преимущества для определенных случаев использования



Wayland

– Новейший протокол сервера отображения для замены X
– Использует клиент-серверную модель с прямым взаимодействием приложений с композитором
– Пространнее архитектура с меньшим количеством компонентов
– Менее сетечно-прозрачен - предназначен в основном для локального использования
– По-прежнему развивается, особенно у NVIDIA для игр
– Wayland устраняет необходимость промежуточного X сервера
– Wayland упрощает стек графических устройств и улучшает безопасность
– Wayland может предложить лучшую производительность и меньший использование ресурсов
– Wayland не имеет API для отрисовки.
– Wayland пишут разработчики X, которые устали исправлять ошибки, удалять костыли, чинить и очищать код [хабр].

Окружения рабочего стола используют подсистемы для создания интерфейса.

Примеры: KDE, MATE, GNOME, Cinnamon, Enlightenment

5) Понятие командного интерпретатора, примеры

Командный интерпретатор – это программа, которая обеспечивает взаимодействие пользователя с операционной системой через интерфейс командной строки.

Шелл – это командная оболочка, запускаемая при входе пользователя в систему и предоставляющая средства работы с ОС.

Часто в Linux шелл = командному интерпретатору

В скриптах можно явно указать, в каком интерпретаторе выполнять скрипт с помощью шабанг: *#!/bin/bash*, для пользователя запускаемый интерпретатор указывается в */etc/passwd*

Примеры командных интерпретаторов

Bash (Bourne Again Shell) был разработан Брайаном Фоксом для проекта GNU в 1989 году. Он был призван стать свободной альтернативой оболочке Bourne (sh), откуда и берет своё имя. Стал основным шеллом для большинства дистрибутивов Linux и macOS. Bash популярен, прост, поддерживает скриптинг, автодополнение команд, имеет расширения.

Zsh (Z Shell) был разработан Полом Фалстадом начиная с 1990 года. Изначально задумывался как альтернатива и улучшение Bash. Zsh умеет все то, что и Bash. Но он гибок, есть темы, плагины, шоткаты.

Sh (Bourne Shell) - базовый интерпретатор.

А также

- Fish (Friendly Interactive Shell),
- Tcsh и Csh - более старые интерпретаторы, менее популярны сейчас
- Ksh (Korn Shell) - сочетание возможностей Bourne и C shells.
- Posh - PowerShell-подобный shell для Linux.

6) Вызов команд и утилит, передача параметров, стандарт POSIX

Структура команд Linux: ***команда [опции] [аргументы]***

Команда: имя программы или утилиты, которую нужно выполнить. Оболочка проверяет доступность этой команды, обычно, обращаясь к переменной окружения PATH, которая содержит список каталогов, где могут быть найдены исполняемые файлы.

Аргументы: данные, передаваемые команде для ее работы. Аргументы могут быть файлами, каталогами, строками или другими значениями, зависящими от конкретной команды.

***Опции*:** параметры, которые изменяют поведение команды. Они обычно начинаются с **-p** или **--option.** Иногда у опции может быть значение **–key=value**

Специальные символы

- **“Кавычки”** необходимы для передачи аргументов, содержащих пробелы, специальные символы или переменные
- *** (звездочка):** Маска для подстановки множества файлов.
- **? (вопрос):** Маска для подстановки одного символа.
- **> (больше):** Перенаправление стандартного вывода в файл.
- **< (меньше):** Перенаправление стандартного ввода из файла.
- **| (пайп):** Передача стандартного вывода одной команды на стандартный ввод другой.
- **& (и):** Запуск команды в фоновом режиме.

Параметры вводятся друг за другом слева направо, разделённые пробелом.

POSIX (Portable Operating System Interface) – это семейство стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов.

POSIX стандартизирует:

- Имена и поведение команд.
- Синтаксис командной строки.
- Обработку ошибок.
- Взаимодействие с файловой системой.

Стандартный синтаксис командной строки определен во 2ой части стандарта POSIX и поддерживается всеми командными интерпретаторами.

7) Получение справки в Linux, man, страницы man

Получение справка

- Встроенная справка **–h** или **–help** опция после команды
- **man [команда]** – страницы с информацией о команде
- **tldr [команда]** – сокращённый man

Обозначения

- **[ ]** – указывают необязательные опции/параметры команды
- **|** – указывает альтернативность опций - XOR
- **{ }** – выбор из вариантов - один обязательно

Прочие утилиты

- **type** - позволяет определить типы команд
- **which** – где лежит исполняемый файл

Команда ***man man*** откроет справку про man. Выход **q**, информация про клавиши управления **h**.

Структура страницы man обобщённая

NAME	Имя	Начало страницы, содержит номер раздела и номер страницы, а также название команды или функции
DESCRIPTION	Описание	Краткое описание того, что делает команда
SYNTAX	Синтаксис	Показывает правильный синтаксис использования команды
OPTIONS	Параметры	Описание каждого параметра команды и их значения
RETURN VALUES	Возвращаемые значения	Описание возможных выходных значений команды
EXAMPLES	Примеры	Конкретные примеры использования команды
ERRORS	Диагностика ошибок	Описание возможных ошибок и их решений
ADDITIONAL NOTES	Примечания	Дополнительная информация, которая может быть полезна
HISTORY	История	Информация о версии команды и ее истории
AUTHOR	Авторские права	Указание на авторов или лицензию
SEE ALSO	Сайты	Ссылки на другие страницы man, которые могут быть полезны

Страницы /usr/share/man/man[0-9]

1	General (User) Commands	Стандартные команды
2	System Calls	Системные вызовы
3	Library Calls	Функции библиотек С
4	Special Files	Специальные файлы
5	File Formats and Conventions	Специфичные файлы
6	Games	Игры
7	Miscellaneous	Подробное руководство
8	System Administration Commands	Системные команды
9	Kernel Routines	Локальные команды

8) Bash. Работа с переменными

Все переменный одного типа – **строка**. Переменные могут хранить строки, числа, массивы и другие типы значений. Bash интерпретирует данные в зависимости от контекста их использования.

ПРАВИЛА ИМЕНОВАНИЯ

Переменные могут начинаться с буквы (как строчной, так и заглавной) или символа подчеркивания (_). После первого символа могут следовать буквы, цифры или символ подчеркивания. Bash чувствителен к регистру.

ОБЪЯВЛЕНИЕ

VARIABLE_NAME="value"

VARIABLE_NAME=5

ОБРАЩЕНИЕ

$VARIABLE_NAME

${VARIABLE_NAME}

ОПЕРАЦИИ С ДАННЫМИ

Результат команды *variable=$(command --key)*.

Результат целочисленных операций *variable=$((...))*.

Для работы с дробными числами *bc*.

ОСОБЫЕ ПЕРЕМЕННЫЕ

$0, $1, ${10} и т.д.

$EUID – эффективный UID.
$UID –содержит реальный идентификатор, который устанавливается только при логине.
$GROUPS – массив групп к которым принадлежит текущий пользователь
$HOME – домашний каталог пользователя
$HOSTNAME – hostname компьютера
$HOSTTYPE – архитектура машины.
$PWD – рабочий каталог
$OSTYPE – тип ОС
$PATH – путь поиска программ
$PPID – идентификатор родительского процесса
$SECONDS – время работы скрипта (в секундах)
$- – общее количество параметров, переданных скрипту
$* – все аргументы, переданные скрипту (выводятся в строку)
$@ – то же самое, что и предыдущий, но параметры выводятся в столбик
$! – PID последнего запущенного в фоне процесса
$$ – PID самого скрипта

9) Bash. Потоки ввода-вывода

Стандартный ввод (/dev/stdin, 0):

Поток ввода данных для программы. Читает информацию из клавиатуры или из файла или вывода другой команды.

Стандартный вывод (/dev/stdout, 1):

Поток вывода данных результатов работы. Пишет в терминал или перенаправляется в файл.

Стандартный поток ошибок (/dev/stderr, 2):

Поток вывода сообщений об ошибках и других диагностических сообщений. Идет в терминал, но может быть перенаправлен в файл или другой поток.

Перенаправление в файл:

- **>** вывод (**>>** добавление),
- **2>** ошибок,
- **&>** вывода и ошибок,
- **| (пайп):** перенаправляет вывод одной команды на ввод другой,
- 2>&1 поток stderr перенаправляется в тот же поток, что и stdout.

Пустой файл **/dev/null,** рандом **/dev/random, /dev/urandom,** файл с нулями **/dev/zero**

10) Bash. Конвейеры (|, ||, &&, >, >>, <)

Символ	**Значение**	**Пример**
| (Пайп)	Передает вывод одной команды в качестве входных данных другой команде	ls -l | grep keyword
|| (ИЛИ)	Выполняет вторую команду только если предыдущая команда неудачно завершила работу	mkdir new_directory || echo "Command failed"
&& (И)	Выполняет вторую команду только если предыдущая команда успешно завершила работу	mkdir new_directory && echo "Directory created successfully"
> (Перенаправление вывода)	Перенаправляет вывод команды в указанный файл	echo "Hello, World!" > output.txt
>> (Добавление вывода в файл)	Добавляет вывод команды к существующему файлу	echo "New line" >> existing_file.txt
< (Перенаправление ввода)	Указывает источник ввода для команды	sort < input.txt

11) Bash. Условные операторы

Операторы **elif** и **else** необязательные. Операторы **then** и **fi** отделяются новой строкой или “;”.

Виды условий (пробелы важны):

- [условие]
- [[более гибкое условие]]
- ((арифметическое выражение))

! – отрицание

```
if [ $num -gt 0 ]; then
  echo "Число положительное"
elif [ $num -lt 0 ]; then
  echo "Число отрицательное"
else
  echo "Число равно нулю"
fi
```

Сравнение чисел

-eq – равно,

-ne - не равно,

-lt – меньше,

-le - меньше или равно,

-gt – больше

-ge - больше или равно

Сравнения строк:

-z - строка пуста

-n - строка не пуста

= или == - строки равны

!= - строки не равны

< - меньше (сравниваются коды символов)

<= - меньше или равно (сравниваются коды символов)

> - больше (сравниваются коды символов)

>= - больше или равно (сравниваются коды символов)

Полезные флаги для проверки файлов

- -e: файл существует (равноценный -a)
- -f: файл существует и является обычным файлом (не директория)
- -r: файл существует и имеет права на чтение
- -w: файл существует и имеет права на запись
- -x: файл существует и имеет права на выполнение
- -ef файл2: проверяет, что оба файла являются жесткими ссылками на один и тот же файл

Оператор **case in**

```
case word in
  pattern1)
    # действия для pattern1
    ;;
  pattern2)
    # действия для pattern2
    ;;
  *)
    # действия для остальных случаев
    ;;
esac
```

```
#!/bin/bash

day=$1

case $day in
  Monday|Tuesday|Wednesday|Thursday|Friday)
    echo "Это рабочий день"
    ;;
  Saturday|Sunday)
    echo "Это выходной"
    ;;
  *)
    echo "Неверный ввод"
    exit 1
    ;;
esac
```

12) Bash. Циклы

For loop

```
for var in list; do
  commands
done
```

Пример:

```
fruits=(apple banana cherry)
for fruit in "${fruits[@]}"; do
  echo "Фрукт: $fruit"
done
```

Пример 2:

```
for i in $(seq 1 100); do
  echo $i
done
```

C-синтаксис for

```
for (counter=1; counter < 10; counter++);  do
   commands
done
```

While loop (пока условие верно)

```
while condition; do
  commands
done
```

Пример:

```
i=1
while [ $i -le 5 ]; do
  echo $i
  ((i++))
done
```

Until loop (пока условие НЕверно)

```
until condition; do
  commands
done
```

Пример:

```
i=1
until [ $i -gt 10 ]; do
  echo $i
  i=$((i+1))
done
```

13) Bash. Работа с файлами (ввод - вывод)

Перенаправление и запись:

- > перенаправляет stdout в файл (перезаписывает).
- >> перенаправляет stdout в файл (добавляет).
- < перенаправляет stdin из файла.
- 2> перенаправляет stderr в файл (перезаписывает).
- 2>> перенаправляет stderr в файл (добавляет).
- >& или &> перенаправляет stdout и stderr в файл.
- tee: вывод в несколько файлов

Чтение

- cat: Выводит содержимое файла.
- head/tail: Выводит начало/конец файла.

Команды для работы с файлами

- touch: Создает пустой файл или обновляет время.
- cp/mv/rm: Копирование, перемещение, удаление.
- mkdir/rmdir: Создание/удаление каталогов.

Поиск и обработка

- wc: Подсчет строк, слов, символов.
- grep: Поиск по шаблону.
- sed/awk: Текстовая обработка.
- find: Поиск файлов/директорий.

14) Bash. Функции.

```
function_name() {
  local a = $1 # объявление локальной переменной
  echo “$2”
  return 0 # код результата от 0 до 255
}

function_name аргумент1 аргумент2 # вызов с передачей параметров
```

15) Типы файлов в Linux

Утилита **file** определяет тип файла по его содержанию.

Команда **ls -l** отображает все доступные параметры файла: первая буква – тип файла.

-	обычный файл
d	каталог
l	символическая ссылка
b	блочное устройство (/dev)
c	символьное устройство (/dev)
p	именованный канал
s	сокет

Обычные файлы (Regular files) «—»

Эти файлы содержат данные, текст, программы или другие данные. Они могут быть любого типа, например, текстовыми, бинарными, изображениями и т.д.

```
-rw-r--r-- 1 root root  15322 Dec 10 21:33 alt.log
```

Каталоги (Directories) «d»

Это специальный тип файлов, который содержит список других файлов и подкаталогов.

```
drwxr-xr-x 1 root root   4096 Jul 19 06:52 apt
```

Символьные ссылки (Symbolic links)

– ***Мягкие*** (символические) ссылки **ln -s** – отдельные файлы, указывающие на путь к другому файлу или директории, похоже на «ярлыки». Обозначение: **«l».**
– ***Жесткие*** ссылки **ln** – альтернативные имена. Обозначение**: «—».**

Зачем?

– Упрощение доступа к файлам,
– Перенаправление на актуальные файлов для совместимости
– Разграничение доступа

В чем разница?

– Жесткая ссылка в пределах одной файловой системы в одном томе
– Мягкая может ссылаться на другой том, файловую систему, компьютер
– Мягкая может быть на каталог
– На жесткую можно назначить отдельные права

Специальные файлы устройств (Device files)

В Linux доступ к устройствам, таким как жесткие диски, порты и устройства ввода/вывода, осуществляется через специальные файлы устройств (ls –la /dev).

– ***Блочные устройства (Block devices)*** «**b**»: Устройства, доступ к которым осуществляется блоками данных (например, жесткие диски).
– ***Символьные устройства (Character devices)*** «**c**»: Устройства, доступ к которым осуществляется посимвольно (например, клавиатура, мышь).

Файлы каналов (Pipes) «p»

– Это файлы, используемые для передачи данных между процессами.
– Именованные каналы (Named pipes, FIFO): файлы, которые позволяют процессам общаться друг с другом, даже если они не связаны напрямую.

Зачем?

– Межпроцессное взаимодействие (IPC). FIFO позволяет запустить несколько процессов, где одни процессы записывают данные, а другие их читают. Это полезно для создания программ, которые взаимодействуют друг с другом в реальном времени.
– Конвейеризация процессов.
– Упрощение взаимодействия между программами, написанными на разных языках программирования или выполняться как отдельный процесс.

Сокеты (Sockets) «s»

Это файлы, используемые для сетевых соединений между процессами. Обычно используются для межпроцессного взаимодействия или сетевых соединений.

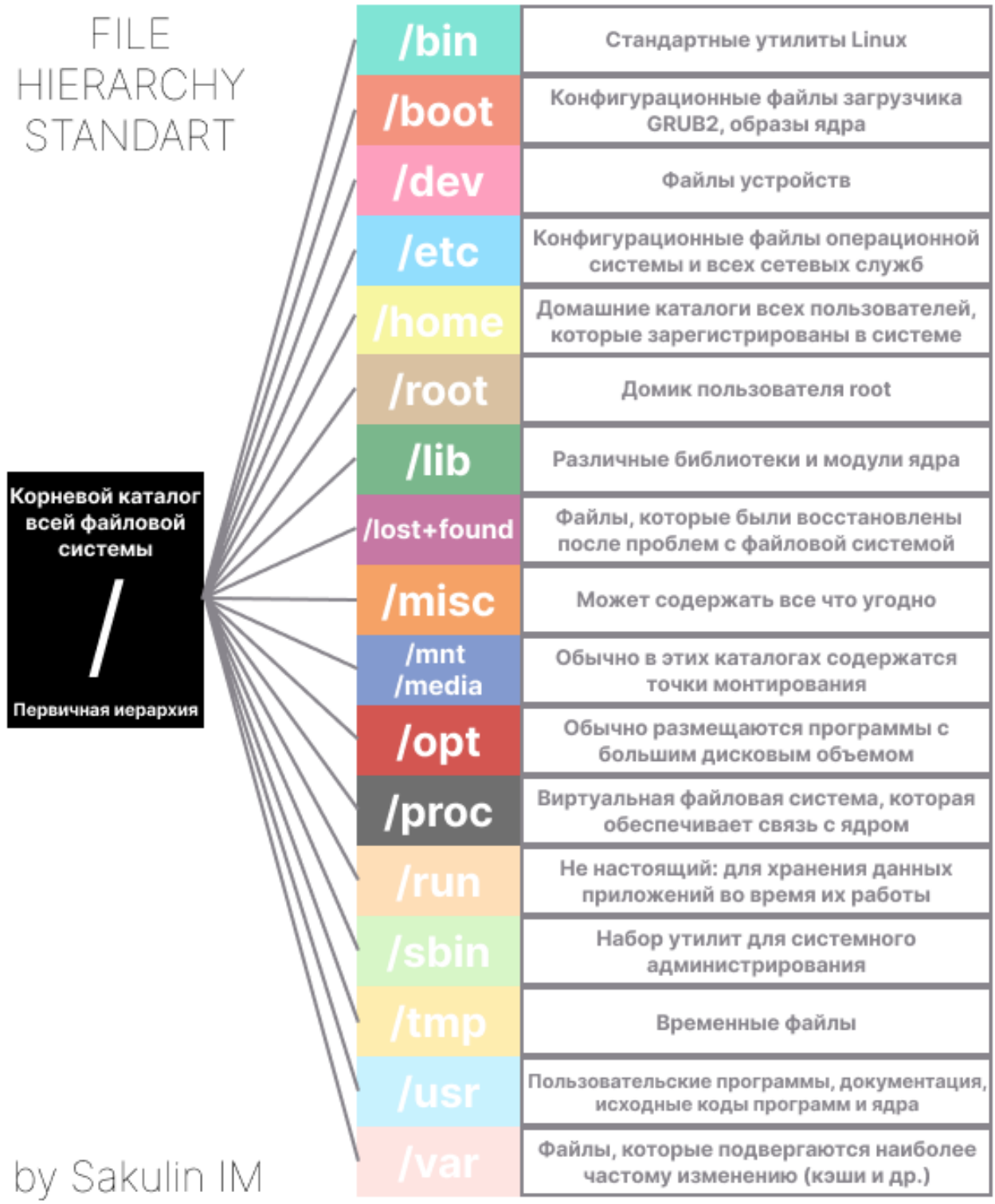
Файлы ядра (Special Kernel Files)

Это файлы, которые представляют собой виртуальные устройства и интерфейсы с ядром системы, например, файлы, которые находятся в каталогах /proc и /sys.

16) Filesystem Hierarchy Standard (FHS), структура и назначение каталогов

Стандарт на структуру каталогов в Linux называется **Filesystem Hierarchy Standard (FHS).**

Этот стандарт определяет расположение и назначение файлов и каталогов в Unix-подобных операционных системах, включая Linux. FHS помогает поддерживать согласованность и совместимость между различными дистрибутивами Linux и Unix-системами, определяя, где должны находиться различные типы файлов и для чего они предназначены.



17) Каталог /proc

Каталог /proc – это особая виртуальная файловая система, которая не хранится на диске или в оперативной памяти. Ее содержимое генерируется ядром на лету при запросе, но работает настолько прозрачно, что кажется обычным файловым каталогом. В ней находятся файлы с информацией о системе и процессах, доступные для чтения любыми редакторами, включая параметры ядра, которые можно изменять в /proc/sys.

Использование /proc:

- **Отладка программ:** отслеживания поведения, потребления ресурсов, открытых файлов и т. д.
- **Мониторинг системы:** информация о процессоре, памяти, загрузке, версии ядра, подключенных файловых системах, процессах и системных ресурсах.

Основные директории

- /proc/[pid]: Подкаталог для каждого процесса, где [pid] - идентификатор процесса. Содержит информацию о конкретном процессе.
- /proc/self: Символическая ссылка на текущий процесс пользователя.
- /proc/net: Информация о сетевых интерфейсах и протоколах.
- /proc/stat: Статистика использования CPU, ввода-вывода и других ресурсов.
- /proc/version: Версия ядра Linux.
- /proc/cpuinfo: Информация о процессорах.
- /proc/meminfo: Статистика использования памяти.

Дополнительные директории

- /proc/filesystems: Список поддерживаемых файловых систем.
- /proc/partitions: Список разделов на устройствах.
- /proc/ioports: Информация о портах ввода-вывода.
- /proc/interrupts: Прерывания и количество обработанных ими событий.
- /proc/kallsyms: Символические имена функций ядра.

18) Файловая система на примере EXT4

Файловая система (File System) – механизм определяющий способ организации, хранения и именования данных. Содержит структуру и механизмы работы с именованными данными.

- Extended Filesystem — является стандартом для Linux.
- Удобна для большого количества маленьких файлов
- Размер файла до 16 ТиБ (2^40)
- Размер ФС до 1 ЭиБ (2^60)
- Обратная совместимость: EXT4 совместим с EXT3
- Улучшенное распределение (распределение блоков хранения перед записью на диск)
- Контрольные суммы журнала (выполняются проверки для выявления ошибок в объеме блока)
- Неограниченное количество подкаталогов (за исключением самого размера каталога)
- Прозрачное шифрование (с ядра Linux 4.1).

Журналирование

EXT4 записывает изменения в специальный журнал перед их применением к основной файловой системе. Это позволяет быстро восстановить файловую систему после сбоя питания или аппаратных проблем, так как система может прочитать журнал и откатить изменения, которые не были успешно применены.

EXT4 поддерживает **три режима ведения журнала**:

- Journal - записываются данные и метаданные.
- Ordered - записываются только метаданные, а данные записываются после метаданных.
- Writeback - записываются только метаданные, а данные могут быть записаны в любое время.

Запись в каталоге - содержит имя файла и номер индексного дескриптора - inode (ссылку на дескриптор - inode). Фактически запись о файле в каталоге – это ссылка на inode.

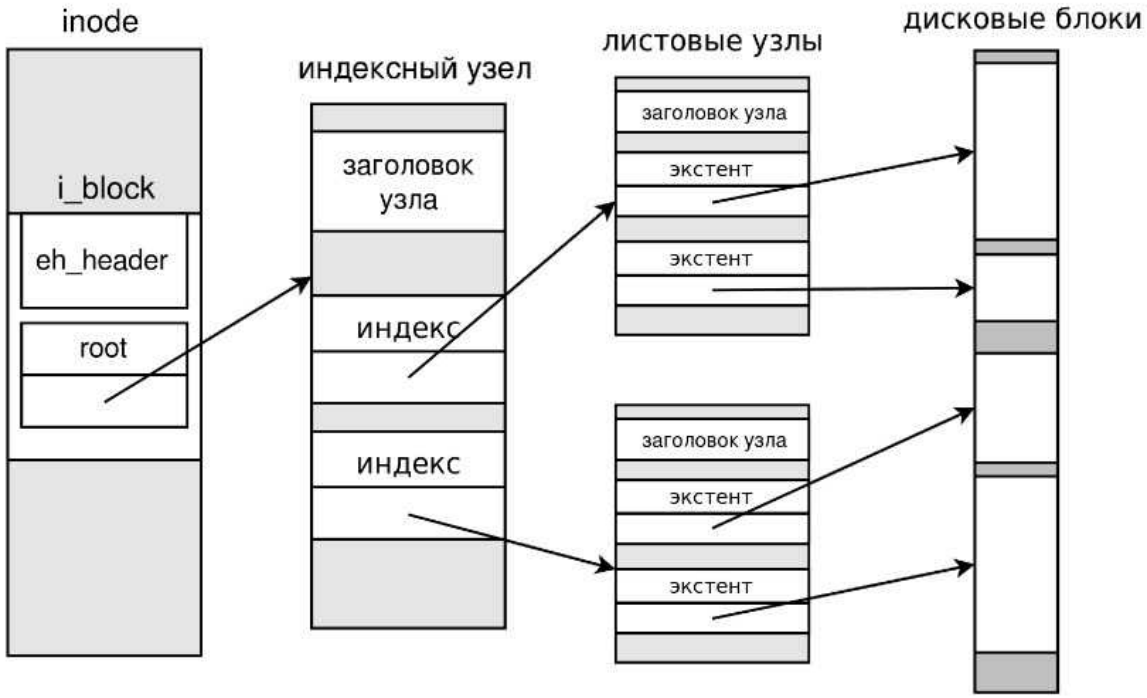
Индексный дескриптор – inode – метаданные файла, содержащие всю описательную информацию о файле и ссылки на блоки с данными.

Метаданные в inode:

- Имя файла или директории
- Тип файла (регулярный, директория, символическая ссылка и т.д.)
- Права доступа
- UID и GID владельца
- Временные метки
- Ссылка на расположение данных

Блоки с данными файла - Непосредственно полезные данные. Хранится в спец. области файловой системы, часто ограниченной по размеру.

Эксент (extent) — это непрерывная область на диске, отведенная под хранение данных файла.



Основные компоненты файловой системы EXT4

- Суперблок (Superblock): содержит информацию о файловой системе, такую как размер блока, количество inodes, и другие параметры.
- Группы блоков (Block Groups): разделение файловой системы на логические группы блоков, каждая из которых имеет свой суперблок, таблицу дескрипторов групп и область данных.
- Таблица дескрипторов групп (Group Descriptor Table): содержит информацию о каждой группе блоков.
- Битовая карта блоков (Block Bitmap): отслеживает, какие блоки данных заняты, а какие свободны.
- Битовая карта inodes (Inode Bitmap): отслеживает, какие inode используются, а какие свободны.
- Таблица inodes (Inode Table): содержит информацию об inodes, включая метаданные о файлах и каталогах.
- Область данных (Data Blocks): содержит реальные данные файлов.

19) Монтирование файловой системы. Файл /etc/fstab

Монтирование файловой системы — это процесс подключения файловой системы к определённой точке в иерархии директорий операционной системы, называемой точкой монтирования.

В результате монтирования файловая система становится доступной для работы, и её файлы и каталоги могут быть использованы пользователем или приложениями.

Основные понятия

– ***Файловая система*:** организованный способ хранения файлов на диске или другом носителе (например, EXT4, NTFS, FAT32).
– ***Точка монтирования*:** каталог в иерархии файловой системы Linux, к которому подключается файловая система.
– ***Монтирование (mount):*** процесс связывания файловой системы с точкой монтирования.
– ***Размонтирование (umount):*** процесс отключения файловой системы от точки монтирования.

Управление монтированием

– mount / umount – утилиты для ручного монтирования
– lsblk и blkid – информация о дисках
– Файл /etc/fstab – конфигурация монтирования при старте системе

Файл /etc/fstab

– Файл устройства (или UUID) — указывает на устройство
– Точка монтирования — указывает на каталог монтирования
– Тип файловой системы — определяет тип файловой системы, например, ext4, vfat, ntfs, nfs, и т.д.
– Параметры монтирования — опции, которые определяют, как файловая система будет монтирована (defaults, ro, rw, noexec)
– Параметр дампа — число (часто 0 или 1), которое указывает, должна ли утилита dump делать резервные копии этой файловой системы. Значение 0 означает, что резервное копирование не нужно.
– Параметр проверки файловой системы (fsck) — число, указывающее порядок проверки файловых систем при загрузке системы. Корневая файловая система обычно имеет значение 1, а все остальные — 2 или 0, что означает отсутствие проверки.

Пример /etc/fstab

```
UUID=f930bb52-288a-46c7-9bed-46a84fe710d3 /                       ext4    defaults        1 1
UUID=e3a0fced-9dab-47d8-ad43-355c2dadd8fd swap                    swap    defaults        0 0
/dev/sda5       /media/data             ntfs-3g         defaults        0       0
/dev/sda6       /media/resource         ntfs-3g         defaults        0       0
/dev/sda7       /media/multimedia       ntfs-3g         defaults        0       0
```

20) LVM, основные понятия

LVM (Logical Volume Manager) — это подсистема управления логическими томами, предоставляющая гибкий и мощный способ управления дисковым пространством в Linux. Вместо того чтобы работать с физическими разделами напрямую, LVM создаёт абстрактный слой между физическими дисками и логическими томами, позволяя динамически изменять размер, добавлять и удалять диски без простоев системы.

PV (физический том): физический том, раздел диска, используемый LVM.

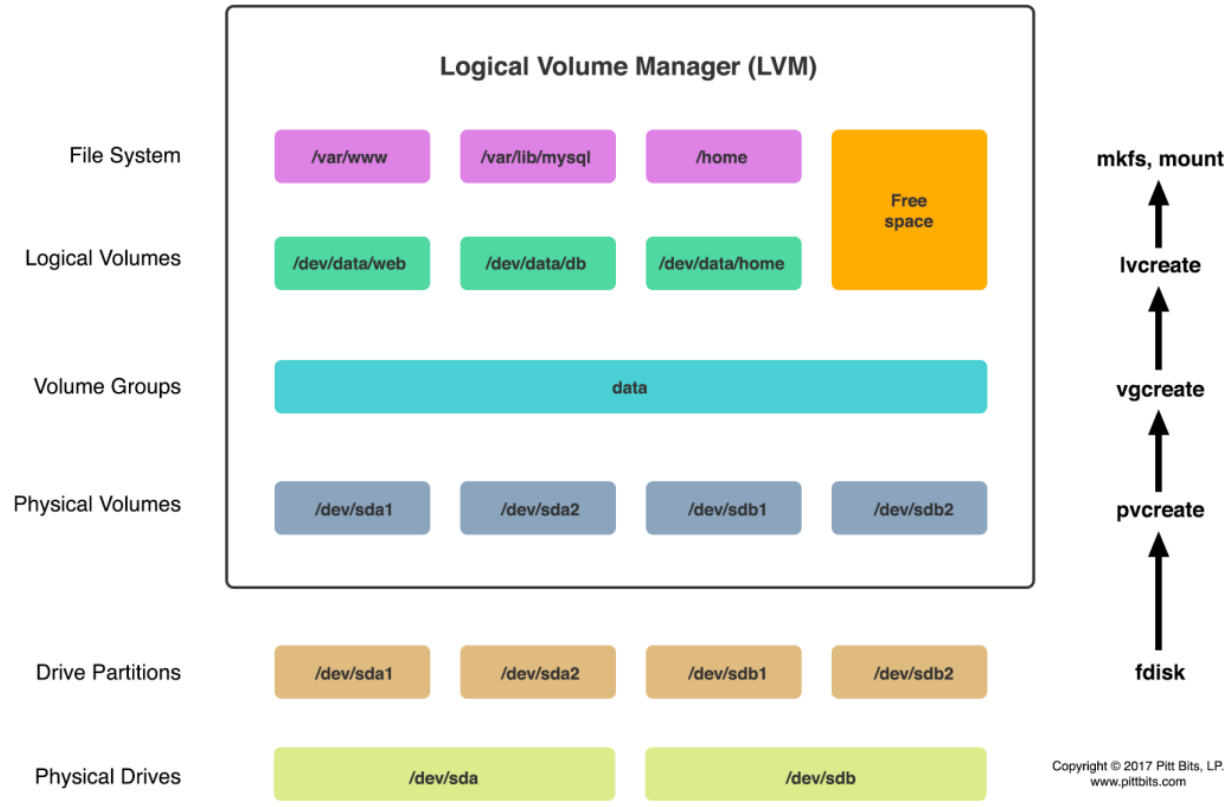
Обычно это раздел на диске или весь диск. В том числе, устройства программного и аппаратного RAID. **PE (физический экстент):** минимальная единица пространства в PV.

VG (группа томов): группа физических томов, общий пул хранения.

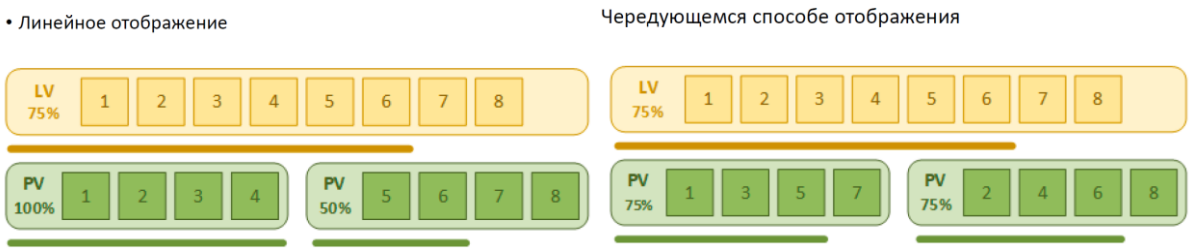
Это самый верхний уровень абстрактной модели, используемой системой LVM. С одной стороны группа томов состоит из физических томов, с другой - из логических и представляет собой единую административную единицу.

LV (логический том): логический том, аналог раздела, созданный из VG.

Раздел группы томов, эквивалентен разделу диска в не-LVM системе. Может содержать файловую систему. **LE (логический экстент) –** минимальная единица в LV, связанная с PE.

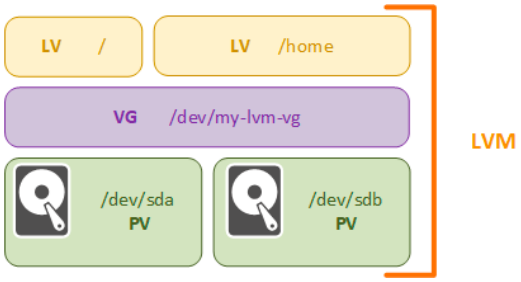


Экстенты и способы их отображения



Преимущества LVM:

- **Гибкость:** Динамическое изменение размеров LV.
- **Динамическое выделение:** Добавление дисков в VG.
- **Снимки:** Возможность создания резервных копий.
- **Управление:** Централизованное управление дисками.



*Самое **главное** и самый большой минус **LVM** — он не читается grub'ом поэтому раздел /boot должен находиться вне **LVM** на отдельном разделе.*

21) Безопасность. Идентификация, Аутентификация, Авторизация.

Идентификация – это процесс, когда система определяет, существует ли конкретный пользователь в ней**.** Необходима для того, чтобы система могла понять, кто именно пытается получить доступ. ОПРЕДЕЛЕНИЕ СУЩНОСТИ

- *Логин пользователя:* Система регистрирует имя пользователя, пытающегося получить доступ.
- *Идентификация процессов:* Каждый процесс имеет уникальные идентификаторы пользователя (UID) и группы (GID).
- *Идентификация устройств:* Устройства идентифицируются по именам в /dev/ и другим характеристикам.

Аутентификация – это проверка подлинности пользователя или устройства, пытающегося получить доступ к ресурсу. Пользователь предоставляет доказательства своей личности. Невозможна без идентификации. ПОДЛИННОСТЬ СУЩНОСТИ

- *Аутентификация по паролю:* система сравнивает введенный пароль с хешем, хранящимся в /etc/shadow или на сервере аутентификации.
- *Аутентификация по ключу (SSH):* используется криптографическая проверка соответствия ключей.
- *PAM (подключаемые модули аутентификации):* модульная система для подключения различных методов аутентификации. Настройки в /etc/pam.d/.
- *Аутентификация процессов:* процессы наследуют UID и GID пользователя.

Авторизация в Linux – это процесс присвоения пользователям определенных прав доступа и привилегий. ПРАВА СУЩНОСТИ

- *Права доступа к файлам и каталогам:* система проверяет права (чтение, запись, выполнение) для владельца, группы и других пользователей. Настройка через chmod и chown.
- *Права суперпользователя (root):* используется команда sudo. Настройка прав в /etc/sudoers.
- *Списки контроля доступа (ACL):* позволяют задавать более детальные права доступа. Управление через setfacl и getfacl.
- *Система AppArmor и SELinux:* принудительный контроль доступа, ограничивающий действия процессов. Настройки в /etc/apparmor.d/ и /etc/selinux/config.

22) Безопасность в Linux. Пользователи. Файлы passwd и shadow

- Linux – многопользовательская система
- Пользователь идентифицируется UID
- Всегда UID root = 0
- UID 1-99 – системные пользователи, UID 1000 и выше для обычных пользователей.

Типы пользователей

- Администраторы (привилегированные) - имеют полный доступ к системе. По умолчанию есть один администратор - пользователь root.
- Локальные пользователи - обычные пользователи с ограниченным доступом. Их учётные записи создаются администратором.
- Системные пользователи - автоматически создаются системой для работы внутренних процессов и служб.

Файл /etc/passwd

ivan:x:1000:1000:ivan,,,:/home/ivan:/bin/bash

Username:Password:UID:GID:GECOS,,,:HomeDirectory:LoginShell

– Username. Строка, которую вы вводите при входе в систему
– Password. В старых системах Linux зашифрованный пароль пользователя хранился в файле /etc/passwd.
– UID. Идентификатор пользователя – это номер, назначенный каждому пользователю.
– GID. Номер идентификатора группы пользователя, относящийся к основной группе пользователя.
– GECOS или полное имя пользователя. Это поле содержит список значений через запятую со следующей информацией:
  - Полное имя пользователя или название приложения.
  - Номер комнаты.
  - Рабочий номер телефона.
  - Домашний телефон.
  - Другая контактная информация.
– Home directory. Абсолютный путь к домашнему каталогу пользователя. Он содержит файлы пользователя и конфигурации. По умолчанию домашние каталоги пользователей именуются по имени пользователя и создаются в каталоге /home.
– Login shell. Абсолютный путь к оболочке входа пользователя. Это оболочка, которая запускается, когда пользователь входит в систему. В большинстве дистрибутивов Linux оболочкой входа по умолчанию является Bash.

Файл /etc/shadow

ivan:6zHvrJMa5Y690smbQ$z5zdL...:18009:0:120:7:14::

– Имя пользователя. Строка, которую вы вводите при входе в систему. Учетная запись пользователя, которая существует в системе.
– **$type$salt$hashed**. Зашифрованный пароль. **$type** является методом криптографического алгоритма хеширования и может иметь следующие значения:

$1 – MD5, $2a – Blowfish, $2y – Eksblowfish, $5 – SHA-256, $6 – SHA-512

– Последнее изменения пароля. Это дата последнего изменения пароля. Количество дней исчисляется с 1 января 1970 года (дата эпохи).
– Минимальный срок действия пароля. Количество дней, которое должно пройти, прежде чем пароль пользователя может быть изменен. Как правило, он установлен на ноль, что означает отсутствие минимального срока действия пароля.
– Максимальный срок действия пароля. Количество дней после смены пароля пользователя. По умолчанию этот номер установлен на 99999.
– Период предупреждения. Количество дней до истечения срока действия пароля, в течение которого пользователь получает предупреждение о необходимости изменения пароля.
– Период бездействия. Количество дней после истечения срока действия пароля пользователя до отключения учетной записи пользователя. Обычно это поле пустое.
– Срок хранения. Дата, когда учетная запись была отключена. Это представляется как дата эпохи.
– Неиспользованный. Это поле игнорируется. Оно зарезервировано для будущего использования.

23) Безопасность в Linux. Группы. Файлы group и gshadow

Группы пользователей в Linux – это механизм для группировки пользователей и управления правами доступа к файлам и ресурсам.

- Позволяют выдавать одинаковые права нескольким пользователям
- Расширяют возможности управления правами доступа по сравнению с отдельным назначением прав для каждого пользователя
- Используются не только для пользователей, но и для программ и процессов
- root – группа суперпользователя
- users - основная группа обычных пользователей
- daemon - для служебных процессов

Файл /etc/group

daemon:x:2:root,bin,daemon

- Имя группы. По умолчанию при создании нового пользователя создается также его группа с таким же именем, как и регистрационное имя пользователя.
- Зашифрованный пароль или символ x, указывающий на использование файла /etc/gshadow;
- Идентификатор группы GID.
- Список членов, разделенный запятыми без пробелов.

Тут важно: одноименный с названием группы пользователь может существовать в системе и входить в эту группу, но не быть перечисленным в этом списке. Сведения о том, что он в неё входит можно получить из /etc/passwd

Файл /etc/gshadow

group_name:password:last_change_date:member_list

- **group_name** – имя группы
- **password** – это поле содержит хеш-пароль для группы.
- **last_change_date** – здесь указывается дата последнего изменения пароля группы.
- **member_list** содержит список пользователей, являющихся членами группы.

24) Права на файлы и каталоги. Расширенный ACL.

Стандартные права: у каждого файла есть владелец и группа владения.

Пользователь, группа, остальные = RWXRWXRWX

Для файла:

- r(read) - чтение файла разрешено
- w(write) - запись файла разрешена, то есть можно его редактировать, переименовывать, удалять.
- x(execute) - исполнение файла разрешено.

Для каталога:

- r(read) - разрешено просматривать содержимое каталога, то есть можно воспользоваться командойls и посмотреть какие файлы и каталоги содержаться в данном каталоге.
- w(write) - используется совместно с атрибутомx(execute). Позволяет удалять и переименовывать файлы в каталоге.

– x(execute) - при использовании совместно атрибутомr(read) позволяет увидеть атрибуты файла, то есть его размер, дату модификации, права доступа. Одним словом, позволяет полноценно воспользоваться командой ls -l. При использовании совместно с атрибутом (write) позволяет перейти в каталог командой cd, удалять и переименовывать файлы.

Формат записи

– **rwxrwxrwx** – каждый символ отвечает за право, если права нет « **–** », например, rw-r-xr--
– **777** – каждая цифра в двоичном коде – права

При изменении: u – user, g – group, o – other, + добавить право, - убрать право.

Примеры: u+rw, go-x…

Утилиты

– chgrp группа файл- смена группы владельцев файла
– chown владелец файл- смена владельца файла
– chown владелец:группа файл- смены владельца и группы владельцев файла
– chmod права_доступа файл- смена прав доступа к файлу

Особые права

– stickybit – удаляем только свои файлы
– setuID – повышение прав при запуске файла
– setgID – повышение прав до привилегий группы при запуске файла, наследование ID для новых файлов и каталогов

ACL (Access Control List) – это расширенный механизм управления правами доступа к файлам и каталогам в Linux. Он позволяет задавать более гибкие права доступа, чем стандартные Unix-права.

Утилита **getfacl** позволяет получить текущие настройки ACL для файла.

Утилита **setfacl опции права файл** позволяет установить настройки ACL для файла.

Основные опции

– -m (--modify): модифицировать ACL
– -x (--remove): удалить права из ACL
– -d (--default): применить права по умолчанию для новых файлов/директорий

Права

u:user_name:rwx, g:group_name:rwx, m::rwx

Примеры

```
# Добавить права для пользователя
setfacl -m u:admin:rwx file.txt
# Удалить права для пользователя
setfacl -x u:john file.txt
# Установить права по умолчанию для директории
setfacl -d -m u::rwx,g::rx,o::rx directory/
```

25) Повышение привилегий. Команды su и sudo.

su (substitute user)

- Используется для смены пользователя (обычно на root) или выполнения команд от имени другого пользователя
- Требует знать пароль пользователя
- Попадаем в среду «подменного» пользователя пока не наберем exit

sudo

- Используется для повышения прав на конкретное действие
- Не нужно знать пароль «подменного» пользователя, требуется ввести текущий пароль
- Гибкие настройки через файл /etc/sudoers (visudo)
- sudo позволяет текущему пользователю выполнять отдельные команды с привилегиями другого пользователя (обычно root), без полной смены пользователя. Оно основано на конфигурационном файле /etc/sudoers, который задает правила доступа.

Файл /etc/sudoers (visudo), **правила**

%name ALL=(ALL:ALL) ALL

- **%name** - имя пользователя или группу, к которой нужно применить правило (имя группы указывается после символа %, имя пользователя без).
- Первое **ALL** означает, что правило применяется ко всем IP-адресам,
- Второе **ALL**, что указанный пользователь может запускать команды в сессии любого пользователя,
- Третье **ALL** означает, что указанный пользователь может запускать команды в любой группе.
- Последнее **ALL** указывает, что эти правила нужно применять ко всем командам.

Alias в sudoers: ***TYPE name = value1, value2...***

- User_Alias - для групп пользователей
- Runas_Alias - для групп пользователей по UID
- Host_Alias - для списка хостов
- Cmnd_Alias - для списка команд и директорий

Общие настройки

- Defaults requiretty: запрещает использование sudo в терминалах без псевдонима [0]
- Defaults !secure_path: отключает использование secure_path при выполнении команд [0]

26) Управление процессами. PID, PPID, nice

PID (Process ID)

- Каждому запущенному процессе в Linux присваивается уникальный идентификатор (Process ID).
- PID используется OS для идентификации и отслеживания процессов.
- Инициализирующему процессу (init) присваивается PID 1.
- Используется в командах управления процессами (например, kill, ps).
- Можно узнать PID процесса с помощью команды pgrep

```
pgrep <имя_процесса>
```

PPID (Parent Process ID)

- PPID – это PID родительского процесса.
- Отображается в столбцах многих утилит для управления процессами (top, ps, htop).
- Позволяет увидеть иерархические отношения между процессами.
- Каждый процесс (кроме init) имеет родительский процесс.

Nice (Priority)

- Nice – это значение приоритета процесса.
- Используется для распределения нагрузки между процессами.
- Значения nice обычно варьируются от -20 (высокий приоритет) до 19/20 (низкий приоритет).
- Команды для изменения nice:

```
# Запуск с определенным nice
nice -n <value> <команда>
# Изменение nice уже запущенного процесса
renice <value> <PID>
```

27) Загрузка ОС. Этапы загрузки.

1. Инициализация BIOS/UEFI
- После включения компьютера BIOS или UEFI начинает работу
- Выполняет POST (Power-On Self Test) для проверки оборудования
- Найдет загрузочное устройство в соответствии с настроенным порядком
- Загрузит первые 16 КБ кода BIOS/UEFI в память

2. Выбор загрузочного устройства
- BIOS/UEFI выбирает первое из списка загрузочных устройств
- Обычно это жесткий диск, но может быть USB-накопитель или сетевое устройство

3. Загрузка загрузчика
- Читается первый сектор (обычно 512 байт) загрузочного устройства
- Это обычно содержимое MBR (Master Boot Record)
- MBR содержит код первого этапа загрузчика (например GRUB)

4. Передача управления загрузчику
- Код первого этапа загрузчика выполняется
- Он загружает следующий этап (обычно второй этап GRUB)
- В UEFI системах может быть загружен EFI стаб

5. Загрузка ядра Linux
- Второй этап загрузчика загружает ядро Linux
- Ядро развертывается в памяти
- Разкомпрессируется с помощью initramfs/initrd

6. Инициализация ядра
- Ядро инициализирует оборудование
- Сканирует устройства
- Монтирует корневую файловую систему
- Настраивает основной режим работы
- Запускает первые процессы

7. Запуск системы инициализации
- Запускается процесс инициализации (обычно systemd)
- Настраивается среда пользователя
- Запускаются службы и демоны

8. Загрузка пользовательского пространства

- Запускаются процессы пользователя
- Демонстрируется графический интерфейс или консоль
- Пользователь может войти в систему

28) Загрузка ОС. Загрузчик на примере GRUB2.

Загрузчик (GRUB2)

- Позволяет выбрать ядро
- Загружает драйвер файловой системы
- Загружает выбранное ядро

Проблемы, решаемые загрузчиком

- Отсутствие гибкости: само ядро ОС нужно было бы настраивать вручную для загрузки, и грузится могла бы ОДНА ОС.
- Сложное и нестандартное железо: иногда для дискового контроллера, дисков и файловых систем нужны драйвера, хранящиеся в файловой системе на диске на дисковом контроллере. Loop.
- Отсутствие инструмента управления загрузкой до загрузки ОС

Процесс загрузки

1. При запуске компьютера **BIOS передает управление загрузчику GRUB2**.
2. GRUB2 загружает **драйверы** устройств
3. **Ищет файл** в /boot/grub/**grub.cfg**, читает **конфигурацию.**
4. Отображает **меню выбора**, где пользователь выбирает ОС или опции.
5. **Загружает** выбранное **ядро** и начальную **файловую систему**
6. **Передает управление** загруженной **ОС**

Настройка GRUB

Файл параметров GRUB /boot/grub/ grub.cfg (или /boot/efi/EFI/grub/grub.cfg для UEFI-систем).

- Правим **/etc/default/grub**
- Обновляем конфигурации GRUB **update-grub**

Каталог /boot

- Хранит файлы загрузчика и его параметры
- Хранит ядра
- Хранит initramfs
- Должен читаться «стандартными средствами», т.е. базовыми драйверами дисков и ФС

Initramfs – специальная файловая система, которая загружается в память компьютера во время процесса загрузки ОС**.** Остается в /run.

Initramfs/boot/initramfs содержит базовый набор компонентов Linux:

- простейший командный интерпретатор,
- стандартные системные каталоги /bin, /lib, /etc и так далее,
- простейший командный интерпретатор,

– драйвера для всех необходимых устройств, включая жесткие диски, файловые системы и другие устройства, что позволяет ядру взаимодействовать с аппаратным обеспечением.
– драйвера файловых систем.

В initramfs есть все необходимое для монтирования ФС и ядро его монтирует по файлу /etc/fstab, от корня «/» кончено.

29) Загрузка ОС. Уровни загрузки ОС.

SysV init	systemd	Значение
runlevel0	poweroff.target	выключение
runlevel1	rescue.target	однопользовательский режим
runlevel2	multi-user.target	многопользовательский режим без графической оболочки
runlevel3	multi-user.target	аналогично runlevel3 в System V
runlevel4	multi-user.target	обычно не используется, можно настроить
runlevel5	graphical.target	многопользовательский режим с графической оболочкой
runlevel6	reboot.target	перезагрузка

Различия систем инициализации (пригодятся в билетах 29-31)

Характеристика	SysVinit	systemd
Определение	Инит-система для управления запуском и завершением процесса загрузки операционной системы	Современная инит-система и менеджер системных служб
Основная концепция	Базируется на уровнях загрузки (runlevels)	Использует целевые единицы (targets) для определения состояния системы
Управление службами	Использует команды service для управления отдельными службами	Использует команду systemctl для управления всеми аспектами системы
Виды служб	Только службы (service)	Разнообразные типы юнитов (service, socket, target, timer и др.)
Достоинства	Простота использования, широкое распространение	Гибкость, параллельное выполнение задач, эффективное управление ресурсами
Недостатки	Ограничен в параллельном выполнении, менее эффективен в управлении ресурсами	Более сложная структура, требует времени на изучение
Конфигурация	Файлы в /etc/inittab и /etc/rc.d/	Юнит-файлы в /etc/systemd/system/ и /usr/lib/systemd/system/
Переключение уровня	Изменение в файле /etc/inittab или через telinit	Использование команд systemctl isolate <target_name>
Просмотр текущего уровня	Команда runlevel	Команда systemctl get-default

30) Система инициализации на примере systemd. Назначение

Система инициализации – это компонент операционной системы, который запускается первым после загрузки ядра и управляет последовательностью запуска служб и процессов до достижения определенного уровня (System V) или в определенный target (systemd).

Systemd

– подсистема инициализации и управления службами в Linux
– создана Леннартом Пёттерингом и Кеем Сиверсом
– заменила init в большинстве дистрибутивов
– это система больших, скомпилированных исполняемых файлов

Назначение systemd

- **Запуск PID 1:** запускается как PID 1 и запускает службы
- **Управление службами:** предоставляет пользовательский интерфейс для управления службами.
- **Базовая настройка системы:** включает инструменты для базовой настройки системы (имя хоста, дата, язык, системные учетные записи, рабочие директории и настройки, настройка сети, синхронизация сетевого времени и др.)
- **Планирование запуска:** обеспечивает планирование запуска процессов (таймеры systemd)
- **Монтирование файловых систем:** проводит монтирование и размонтирование файловых систем с иерархическим уведомлением обеспечивает более безопасное каскадирование.
- **Управление временными файлами:** позволяет создавать и управлять временными файлами, включая их удаление.
- **Управление устройствами**
- **Журналирование:** создаёт журналы для хранения системных логов и включает инструменты для управления этими журналами

31) Система инициализации на примере systemd. Типы unit.

В systemd **unit** — базовый объект, описывающий ресурс или действие. Unit-файлы — конфигурации, определяющие как systemd взаимодействует с ресурсами. Различные типы unit управляют разными аспектами системы.

Расположение Unit-файлов:

- /usr/lib/systemd/system/: системные файлы.
- /etc/systemd/system/: пользовательские файлы.
- /run/systemd/system/: динамически созданные файлы.

Типы unit

Тип unit	Назначение и функции	Примеры unit
service	Запуск исполняемых файлов и управление процессами	sshd.service, NetworkManager.service
socket	Устанавливает сокеты для прослушивания входящих соединений	ssh.socket, dbus.socket
path	Мониторит изменения в файлах/директориях	.path unit для мониторинга изменений в /etc/passwd
timer	Управляет расписанием выполнения других unit	systemd-timer-list timers
target	Определяет набор связанных unit для определенного состояния системы	multi-user.target, graphical.target
device	Обеспечивает доступ к периферийным устройствам	ata-scsi-id-ata-pnp-device-0.device
mount	Управляет монтированием файловых систем	/dev/sda1.mount, /home/user/mountpoint.mount
automount	Автоматически монтирует разделы при обращении к ним	/home/user/.automount
swap	Управляет разделами обмена	swapfile.swap
slice	Группирует другие units для управления ресурсами	system.slice, user.slice
scope	Объединяет группы процессов как единый unit	ssh-agent@.service
swapfile	Управляет файлами обмена вместо разделов	/swapfile.swap

Практические:

1) Работа с файлами и каталогами (создание, удаление, перемещение, чтение, изменение и т.п.)

Утилиты: touch, mkdir, rm, rmdir, mv, cp, ls, cd, pwd, vi, nano, cat, less, more, head, tail, grep, find, file, split, sed, awk.

2) Получение информации о системе (аппаратура, ip адреса, свободное место на диске, имя хоста, версия ядра и системы)

Утилиты: lscpu (cat /proc/cpuinfo), ip addr show (cat /proc/net/tcp), df -h (cat /proc/meminfo | grep MemAvailable), hostname (cat /proc/sys/kernel/hostname), uname -a (cat /proc/version).

3) Настройка хоста (имя, Ip-адрес через конфигурационный файл).

Утилиты: hostnamectl hostname NAME

```
sudo nano /etc/network/interfaces
auto eth0
iface eth0 inet static
  address 192.168.1.100
  netmask 255.255.255.0
  gateway 192.168.1.1
sudo systemctl restart networking.service
```

4) Установка и удаление ПО из репозитария

Утилиты: apt-get, apt

5) Управление пользователями (создание, удаление, изменение пароля)

Утилиты: useradd -m, userdel, passwd, usermod, id, groups

6) Управление службами через system (перезапуск, статус, остановка, запуск, включение-отключение автозапуска)

sudo systemctl start, stop, enable, disable, restart, status

7) Управление файловой системой. Права пользователя.

Утилиты: ls -l, chmod, chown, chgrp

8) Управление файловой системой – создание раздела, создание фс, монтирование (можно без использования LVM).

Утилиты: fdisk, parted, mkfs.ext4, mount, umount, partprobe, e2fsck, resize2fs

9) Управление процессами (сведения о процессе, завершение процесса, изменения приоритета, запуск в фоновом режиме)

Утилиты: ps, top, kill, nice, renice

Запуск процесса в фоновом режиме: добавление амперсанда (&) в конце команды.

10) Ssh. Подключение через ssh по паролю с указанием порта

Утилиты: ssh

11) Ssh. Передача файла с удаленной машины, на удаленную машину.

Утилиты: scp

12) Сведения о подключенных пользователях.

Утилиты: who, whoami