

Инструменты разработчика. Вопросы.

0. Основные команды оболочки и работа в терминале Linux

(a) Основные команды

- pwd
- cd
- ls и её опции -l, -a и -i. Какая информация отображается при вызове ls -l?
- echo и её опции -n и -e
- printf, спецификаторы %s, %c, %d, %x, %f, %8s, %08d и %.2f.
- cat и её опция -n.
- less и её опции -N и -R. Горячие клавиши Пробел, b, q, g, G, n, N, h. Поиск текста в файле
- Текстовый редактор nano
- file
- alias, создание и удаление алиасов. Просмотр всех алиасов. Как сохранить алиас навсегда?
- man, поиск в конкретном разделе
- date, печать даты/времени в форматированном виде (опция +<формат>)

(b) Команды для манипуляции с файлами

- touch
- mkdir и её опция -p
- cp и её опции -r, -a, -t, -i, -u, -n
- mv, переименование файлов с помощью mv
- rm и её опции -r, -f

(c) Путь к файлу

Абсолютный и относительный путь. Сокращение имён директорий:

- | | |
|------|----------|
| • . | • ~ |
| • .. | • ~alice |

(d) Горячие клавиши, используемые в терминале

- Ctrl-C
- Ctrl-D
- Ctrl-Z

(e) Команды для архивации

- gzip и её опции -d и -k
- bzip2 и её опции -d и -k
- tar и её опции -c, -x, -t, -v, -f, -z и -j

(f) Пакетные менеджеры

Пакетные менеджеры apt и dnf. Подкоманды этих пакетных менеджеров:

- install/remove
- update/upgrade
- search

1. Основы git

(a) Системы контроля версий

Что такое система контроля версий? Локальные, централизованные и распределённые системы контроля версий. Отличие git от других систем контроля версий.

(b) Настройка git

Команда git config. Печать всех настроек git. Добавление и удаление настроек. Уровни настроек:

- Системный
- Глобальный
- Локальный

Файлы конфигурации, в которых хранятся эти настройки. Основные настройки:

- `user.name`
- `user.email`
- `core.editor`
- `core.autocrlf`
- `alias.<name>`

(c) **Создание репозитория**

Команда `git init` и её опция `--bare`. Команда `git clone`.

(d) **Работа с файлами и коммитами**

Рабочая директория. Индекс. Команда `git add`. Добавление всех изменений из рабочей директории в индекс. Удаление файлов из индекса. Команда `git rm` и её опция `--cached`. Локальный репозиторий. Коммит. Хэш коммита. Команда `git commit` и её опция `-m`.

(e) **Команды для просмотра информации индекса и локального репозитория**

- `git status`
- `git show`
- `git diff` и её опция `--staged`
- `git log` и её опции `--oneline`, `--graph`, `--all`, `--pretty=format`
- `git log` для поиска в истории, её опции `--since`, `--author`, `--grep`, `-S`
- `git blame`

(f) **Работа с ветками**

Что такое ветка? Основная ветка `main`. Команда `git branch` и её опции `-d`, `-D`, `-m`, `-M`, `-v`, `-r`, `-vv`.

(g) **Адресация коммитов**

Адресация коммитов с использованием хэша. Полный и сокращённый хэш коммита. Адресация коммитов с помощью веток и указателя `HEAD`. Символы `~` и `^`.

(h) **Перемещение по графу коммитов**

Указатель `HEAD`. Команда `git switch` и её опции `-c` и `--detach`. Чем `git switch` отличается от старой команды `git checkout`? Переход на отдельный коммит. Состояние `detached HEAD`. Чем опасно это состояние?

(i) **Слияние**

Слияние веток. Команда `git merge`. Слияние перемоткой (`fast-forward merge`). Когда используется такой вид слияния? Опция `--no-ff`. Конфликты при слиянии. Когда возникают конфликты? Как разрешить конфликт? Команды

- `git merge --abort`
- `git merge --continue`

Как отменить произведённое слияние?

(j) **Перебазирование**

Перебазирование веток. Команда `git rebase`. Отличие перебазирования от слияния. Преимущества и недостатки перебазирования по сравнению со слиянием. Конфликты при перебазировании. Как разрешить конфликт, возникший при перебазировании? Команды:

- `git rebase --abort`
- `git rebase --continue`
- `git rebase --skip`

Как отменить произведённое перебазирование?

2. Продолжение git

a. **Откат состояния**

Команда `git reset` и её режимы `--soft`, `--mixed` и `--hard`. Как меняется рабочая директория, индекс и локальный репозиторий при использовании `git reset` в каждом из этих режимов? Как отменить произведённый откат состояния? Команда `git reflog`, что она показывает? В каких случаях использование `git reset` может привести к безвозвратной потере данных? Команда `git restore` и её опция `--source`. Как восстановить случайно удалённый файл в рабочей директории?

b. Копирование отдельных коммитов

Команда `git cherry-pick`. В чём недостатки использования `git cherry-pick`? Конфликты при копировании коммитов. Команды:

- `git cherry-pick --abort`
- `git cherry-pick --continue`
- `git cherry-pick --skip`

c. Интерактивное перебазирование

Команда `git rebase -i`. Файл `git-rebase-todo`. Команды интерактивного перебазирования:

- | | |
|-----------------------|-----------------------|
| • <code>pick</code> | • <code>squash</code> |
| • <code>reword</code> | • <code>fixup</code> |
| • <code>edit</code> | • <code>drop</code> |

Конфликты при интерактивном перебазировании. Как отменить произведённое интерактивное перебазирование?

d. Типы файлов в git

- Отслеживаемые (tracked)
 - Индексированные (staged)
 - Неизменённые (unmodified)
 - Изменённые (modified)
- Неотслеживаемые (untracked)
- Игнорируемые (ignored)

e. Игнорируемые файлы

Файл `.gitignore` и то, как с его помощью:

- Игнорировать все файлы в репозитории с данным именем
- Игнорировать все директории в репозитории с данным именем
- Игнорировать один конкретный файл
- Игнорировать файлы по шаблону `*`, `?`, `[...]`
- Сделать исключение из игнорирования

Игнорирование пустых директорий. Как заставить `git` не игнорировать пустые директории?

f. Очистка репозитория

На какие типы файлов не действует команда `git reset --hard`? Какие типы файлов не меняются при использовании `git switch`? Команда `git clean` и её опции `-f`, `-d`, `-x`, `-X`, `-n`.

g. Удалённый репозиторий

Удалённый репозиторий. Ремоут (remote). В чём разница между удалённым репозиторием и ремоутом? Команда `git remote`, её опция `-v` и подкоманды `add`, `remove`, `rename`, `set-url` и `show`.

h. Удалённые ветки

Что такое удалённая ветка (remote branch)? Что такое ветка отслеживания (remote-tracking branch)? Какие имена имеют ветки отслеживания в `git`? На что указывают такие ветки? Когда обновляется состояние таких веток? Команда:

- `git branch -a`

i. Работа с удалённым репозиторием

Команды для взаимодействия локального репозитория с удалённым:

- `git push` и её опции `--all`, `--tags`, `-f`, `--force-with-lease`, `-u`, `--set-upstream`.
- `git fetch` и её опции `--prune` и `--tags`.
- `git pull` и её опции `--rebase` и `--ff-only`. Конфликты при использовании `git pull`.

Чем `git pull` отличается от `git fetch`? Перезапись истории в удалённом репозитории. Отмена изменений в удалённом репозитории. Команда `git push -f` и в чём её опасность? Команда `git revert`.

j. Отслеживающие и upstream-ветки

Что такое отслеживающая ветка (tracking branch)? Что такое upstream-ветка (upstream-branch)? Как привязать отслеживающую ветку к upstream-ветке? Какие преимущества даёт такая привязка? Что будет, если не сделать такую привязку? Команды:

- `git branch -vv`
- `git branch --set-upstream-to=origin/main`
- `git branch --unset-upstream`
- `git push -u origin main`

k. Тэги

Чем тэги отличаются от веток? Зачем нужны тэги? Команда `git tag` и её опции `-d`, `-l`, `-a` и `-m`.

l. Pull request

Веб-сервисы для хранения и работы с удалёнными репозиториями. GitHub. GitLab. Форк. Pull request. Merge request.

3. Продвинутый git (не будет на коллоквиуме)

4. Потоки и конвейеры

a. Основные команды, которые часто используются в конвейерах

- `tac`
- `head` и её опции `-n` и `-c`
- `tail` и её опции `-n`, `-c`, `-f` и `-F`
- `uniq` и её опция `-c`
- `sort` и её опции `-n`, `-r`, `-u`, `-h`, `-k` и `-t`.
- `wc` и её опции `-l`, `-w`, `-c` и `-m`

b. Потоки и перенаправление

Стандартные потоки ввода-вывода:

- `stdin`
- `stdout`
- `stderr`

Куда эти потоки направлены по умолчанию? Перенаправление потока в файл для перезаписи. Перенаправление потока в файл для дозаписи. Перенаправление файла в `stdin`. Перенаправление `stderr` в файл. Как перенаправить `stdout` и `stderr` в один файл? Файл `/dev/null`. Что произойдёт, если перенаправить потоки `stdout/stderr` в этот файл? Что произойдёт, если перенаправить файл `/dev/null` в `stdin`?

c. Расширения оболочки

- Шаблон поиска (wildcard/glob pattern). Wildcard-символы:
 - `*` ◦ `[...]`
 - `?` ◦ `[!...]`
- Brace expansion `{...}`.
- Синтаксис подстановки команд `$(...)`.
- Подстановка процесса `<(...)` и `>(...)`. Примеры использования подстановки процесса.

d. Конвейеры

Что такое конвейер оболочки Bash и как его использовать? Примеры простых конвейеров. Код возврата конвейера. Переменная `PIPESTATUS`. Как перенаправить поток `stderr` в конвейер?

e. Команда find

Опции команды `find`:

- `-type` ◦ `-and`, `-or`, `-not`
- `-name/-iname` ◦ `-maxdepth`
- `-size` ◦ `-print0`
- `-mtime` ◦ `-exec`

Два вида синтаксиса `find -exec`:

- Синтаксис `c \;`
- Синтаксис `c +`

Чем различаются эти два вида синтаксиса? Примеры использования `find -exec`.

f. **Команда xargs**

Что делает команда `xargs`? В каких ситуациях эта команда используется чаще всего? Примеры использования команды `xargs`. Опции `-0`, `-I` и `-P`. Преимущества `xargs` перед `find -exec`.

g. **Команда tee**

Примеры использования команды `tee`. Опция `-a`. Использование `tee` вместе с подстановкой процесса. Использование `echo`, `sudo` и `tee` для записи в файл с правами `root`.

h. **Команда grep**

Синтаксис команды `grep`. Опции:

- | | |
|--|---|
| • <code>-i</code> | • <code>-c</code> |
| • <code>-r</code> | • <code>-o</code> |
| • <code>-n</code> | • <code>-w</code> |
| • <code>-v</code> | • <code>-A</code> , <code>-B</code> и <code>-C</code> |
| • <code>-E</code> / <code>egrep</code> | • <code>--color=auto</code> |
| • <code>-l</code> | • <code>-q</code> |

Приведите примеры использования `grep` в следующих ситуациях:

- Поиск строк, содержащих подстроку, в одном файле.
- Поиск строк, содержащих подстроку, рекурсивно во всех файлах директории.
- Поиск файлов, содержащих подстроку, рекурсивно во всех файлах директории.
- Фильтрация вывода команды с помощью конвейера и `grep`.
- Поиск строк в файле или файлах, соответствующих регулярному выражению.

i. **Регулярные выражения в grep**

Два типа регулярных выражений: BRE и ERE. Основные элементы регулярных выражений ERE:

- Обычные символы
- Точка `.`
- Символьные классы `[...]`
- Группировка `(...)`
- Альтернатива `|`
- Квантификаторы:
 - `*` ◦ `{m}`
 - `+` ◦ `{m,}`
 - `?` ◦ `{m,n}`
- Якоря строки: `^` и `$`.

Предопределённые классы:

- | | |
|--------------------------|---------------------------|
| • <code>[:alpha:]</code> | • <code>[:lower:]</code> |
| • <code>[:digit:]</code> | • <code>[:upper:]</code> |
| • <code>[:alnum:]</code> | • <code>[:punct:]</code> |
| • <code>[:space:]</code> | • <code>[:xdigit:]</code> |

5. Пользователи и права доступа Linux

a. **Пользователи**

UID. Суперпользователь `root`. Системные пользователи. Зачем нужны системные пользователи? Какой диапазон UID обычно используется для системных пользователей, а какой для обычных?

b. **Группы пользователей**

Группы. Основная группа пользователя. Дополнительные группы. GID. Команда `id` и её опция `-u`. Команда `groups`.

c. **Домашняя директория**

Стандартное расположение домашней директории. Что обычно хранится в домашней директории? Файлы `~/.bash_profile` и `~/.bashrc`. Директория `/etc/skel`.

d. **Системные файлы, хранящие информацию о пользователях и группах**

- `/etc/passwd`
- `/etc/shadow`
- `/etc/group`
- `/etc/gshadow`

Какая информация и в каком формате хранится в каждом из этих файлов? Как хранится информация о паролях пользователей? Как хранится принадлежность пользователя к основной группе? Как хранится принадлежность пользователя к дополнительным группам? Как можно отличить обычного пользователя от системного по файлу `passwd`? Что такое `/sbin/nologin`? Как можно понять, что пользователь заблокирован по файлу `/etc/shadow`?

e. **Работа с пользователями и группами**

- `useradd` и её опции `-m`, `-d`, `-s` и `G`.
- `usermod` и её опции `-G`, `-aG`, `-d`, `-s`, `-L` и `-U`.
- `userdel` и её опция `-r`.
- `groupadd` и её опция `-r`.
- `groupmod`
- `groupdel`

Создание/удаление основной группы при создании/удалении пользователя. Использование `usermod` для блокировки/разблокировки пользователя.

f. **Работа с паролями**

Использование команды `passwd` для смены собственного пароля. Использование команды `passwd` для смены пароля другого пользователя.

g. **su и sudo**

Переключение на другого пользователя. Команда `su`. Полное и неполное переключение. Выполнение команды от имени другого пользователя. Команда `sudo` и её опции `-u` и `-i`. Какой пароль требуется для команды `su`, а какой пароль – для команды `sudo`? Чем различаются следующие способы переключения на пользователя `root`:

- `su`
- `su -`
- `sudo -i`

Файл `/etc/sudoers`. В каком формате хранятся записи в файле `/etc/sudoers`? Группа `sudo/wheel`. В чём опасность редактирования файла `/etc/sudoers`? Как безопасно редактировать этот файл?

h. **Права доступа**

Владелец файла. Группа-владелец. Может ли владелец файла не входить в группу-владелец этого же файла? Права доступа. Просмотр прав доступа с помощью `ls`. Символьное и числовое представление прав доступа. Конвертация из одного представления в другое. Команды:

- `chmod` и её опция `-R`. Как можно изменять права пользователя с помощью этой команды, используя символьное и числовое представление прав? Кто и в каких случаях может изменять права доступа?
- `chown` и её опция `-R`. Как изменить группу владельца, используя `chown`? Кто и в каких случаях может изменять владельца файла?
- `chgrp` и её опция `-R`. Кто и в каких случаях может изменять группу владельца файла?
- `newgrp`.

За что отвечают права `r`, `w`, `x` для обычных файлов? За что отвечают права `r`, `w`, `x` для директорий? Права символических ссылок.

i. **Маска**

Права только что созданного файла. Права копии файла. Команда `umask`.

j. **SUID, SGID и Sticky Bit**

За что отвечает SUID? Примеры системных файлов, которые имеют SUID бит. Работает ли SUID на скриптах? За что отвечает SGID? SGID для обычных файлов и для директорий. За что отвечает Sticky Bit. Пример системной директории, которая имеет Sticky Bit. Символьное и числовое представление прав доступа вместе с данными битами.

6. Диски и файловые системы

a. Единицы измерения объёма информации

Десятичные и двоичные единицы измерения.

b. Метаданные файлов

Что обычно включает в себя метаданные файла? Где хранятся метаданные? Просмотр метаданных с помощью команды `ls -l`. Команда `stat` и её опция `-c`. Плейсхолдеры для форматированного вывода команды `stat -c`:

- | | | | |
|------|------|---------|------------|
| • %n | • %s | • %u %U | • %a %A |
| • %i | • %F | • %g %G | • %x %y %z |

Временные метки `atime`, `mtime` и `ctime`. Изменение временных меток с помощью команды `touch`. Опции команды `touch`: `-a`, `-m` и `-d`.

c. Типы файлов

- Обычный файл
- Директория
- Символическая ссылка
- Файл блочного устройства
- Файл символьного устройства

Как узнать тип файла, используя команды `ls`, `stat` и `file`?

d. Файлы устройств

Директория `/dev/`. Файлы устройств как интерфейсы. Какие данные и метаданные хранят в себе файлы устройств? Major и minor number.

i. Файлы символьных устройств

Примеры символьных устройств. Как символьные устройства передают данные? Символьные псевдоустройства `/dev/null`, `/dev/zero`, `/dev/random`.

ii. Файлы блочных устройств

Примеры блочных устройств. Как блочные устройства передают данные? Блочные устройства дисков и разделов. Размер блока блочных устройств. Физический и логический размеры блоков.

e. Разделы

Зачем нужны разделы? Блочные файлы разделов. Команда `lsblk`. Таблицы разделов:

i. Таблица разделов MBR

Где находится таблица MBR и сколько места занимает? Что содержится внутри MBR? Ограничение на количество разделов и размер каждого раздела. Типы разделов: первичный (primary), расширенный (extended), логический (logical).

ii. Таблица разделов GPT

Где находится таблица GPT и сколько место занимает? Что содержится внутри GPT? Ограничение на количество разделов и размер каждого раздела.

Команда `parted -l`. Как узнать, какая таблица разделов используется на диске?

f. Создание разделов с помощью программы parted

- | | |
|-----------|--------------|
| • print | • rm |
| • mklabel | • resizepart |
| • mkpart | • unit |

Запуск `parted` в скриптовом и тихом режиме.

g. Файловые системы

Что такое файловая система? UUID файловой системы. Создание файловой системы на разделе. Команда `mkfs`. Команды для отображения информации о файловых системах:

- `lsblk -f`
- `blkid`
- `df` и её опции `-h`, `-T` и `-i`
- `du` и её опции `-h` и `-s`

h. Монтирование файловой системы

Что такое монтирование файловой системы? Точка монтирования. Команда `mount` и её опция `-o`. Команда `umount`. Опции монтирования:

- `defaults`
- `rw`
- `ro`
- `noexec`
- `nosuid`
- `noatime`

Автоматическое монтирование файловых систем. Файл `/etc/fstab`. Поля в этом файле:

- `file system`
- `mount point`
- `type`
- `options`
- `dump`
- `pass`

Команда `mount -a`. Просмотр точек монтирования с помощью `findmnt`.

i. Команда `dd`

Распространённые варианты использования команды `dd`:

- Создание файлов определённого размера, заполненных нулевыми или случайными байтами
- Копирование файлов
- Просмотр байт файлов
- Просмотр байт дисков/разделов
- Копирование разделов
- Создание образа диска или раздела (файла `.img`)
- Восстановление диска/раздела из образа
- Запись ISO-образа на флешку

Опции команды `dd`:

- `if`
- `of`
- `bs`
- `count`
- `skip`
- `seek`
- `conv=noerror`
- `conv=sync`
- `conv=notrunc`

На что влияет значение размера блока (`bs`) в команде `dd`? Какой `bs` используется по умолчанию?

j. Жёсткие ссылки

Чем является жёсткая ссылка? Команда `ln`. Что происходит при создании жёсткой ссылки? Равноправность жёстких ссылок. Зачем нужен счётчик жёстких ссылок в `inode`?

- Что будет, если удалить исходный файл, на который указывает жёсткая ссылка?
- Жёсткие ссылки на директории (`.` и `..`). Можно ли создать жёсткую ссылку на директорию?
- Можно ли создать жёсткую ссылку на файл, находящийся в другой файловой системе?

k. Символические (мягкие) ссылки

Чем является символическая ссылка? Команда `ln -s`. Как узнать, куда указывает символическая ссылка? Команда `readlink`.

- Что будет, если удалить исходный файл, на который указывает символическая ссылка?
- Можно ли создать символическую ссылку на директорию?
- Можно ли создать символическую ссылку на файл, находящийся в другой файловой системе?

Работа стандартных команд с символическими ссылками:

- Команда `cp` копирует саму ссылку или файл на который она указывает? А команда `cp -a`?
- Команда `chmod` применяется к ссылке или к файлу на который она указывает?
- Команда `rm` применяется к ссылке или к файлу на который она указывает?
- Переходит ли `find` по символическим ссылкам на директории? Опция `-L` команды `find`.

l. Строение файловой системы `ext4` (основы)

Блоки. Суперблок. Группы блоков. Таблица `inode`. Что хранится в `inode`-ах файлов разных типов? Переполнение таблицы `inode`. Журналирование.

m. Распространённые файловые системы

`ext4`, `xfs`, `FAT`, `NTFS`. Виртуальные файловые системы. Особенности каждой из этих файловых систем.

7. Язык Bash

a. Интерпретатор Bash

Что такое терминал? Что такое оболочка? Оболочки `sh` и `bash`. Скрипты Bash. Шебанг.

b. Переменные Bash

Создание и использование переменных в `bash`. Команда `unset`. Переменные среды. Команда `export`. Команда `source`. Как задать переменную среды для одной команды? Как создать свою переменную среды? Файл `~/.bashrc`.

c. Команды оболочки

Внутренние и внешние команды оболочки. Команда `type -a`. Команда `which`. Переменная среды `PATH`. Как создать свою команду?

d. Кавычки

Разница между одинарными и двойными кавычками в `bash`.

e. Аргументы

Аргументы командной строки. Как запустить скрипт, передав ему некоторые аргументы? Переменные `$1`, `$2`, `$0`, `$*`, `$@`. Чем отличается `"$"` от `"$@"`? Проход по аргументам в цикле. Команда `shift`.

f. Коды возврата

Какой код возврата возвращается при успешном завершении программы? Как получить код возврата последней выполненной команды? Коды возврата скобочных команд. Команды `true` и `false`.

g. Условия if

Сравнение строк. Сравнение чисел. Операции проверки:

- | | | | |
|-------------------|-------------------|-------------------|-------------------|
| • <code>-n</code> | • <code>-e</code> | • <code>-d</code> | • <code>-w</code> |
| • <code>-z</code> | • <code>-f</code> | • <code>-r</code> | • <code>-x</code> |

Разница между `[[...]]` и `[...]`. Проверка на совпадение с регулярным выражением `=~`.

Использование операторов `&&` и `||` вместо условий `if`.

h. Группировка команд и подоболочки

i. Управляющая конструкция case esac

Примеры конструкций `case esac`. Использование `glob`-шаблонов в этих конструкциях.

j. Циклы

Циклы `while` и `for`. Примеры циклов. Итерация от нуля до некоторого числа. Итерация по всем элементам из заданного набора. Итерация по всем файлам в директории. Итерация по всем аргументам/элементам массива. Переменная `IFS`. Перенаправление из цикла в файл и наоборот.

k. Манипуляции со строками

Что означают следующие выражения:

- | | |
|---------------------------------|------------------------------------|
| • <code>\$var</code> | • <code>\${var#pattern}</code> |
| • <code>\${var}</code> | • <code>\${var##pattern}</code> |
| • <code>\${#var}</code> | • <code>\${var%pattern}</code> |
| • <code>\${var:N}</code> | • <code>\${var%%pattern}</code> |
| • <code>\${var:N:K}</code> | • <code>\${var/pattern/str}</code> |
| • <code>\${var:-default}</code> | • <code>\${var^^}</code> |
| • <code>\${var:=default}</code> | • <code>\${var,,}</code> |

l. Работа с целыми числами

Арифметические выражения `((...))`. Чем `$((...))` отличается от `((...))`?

m. Функции

Создание функции. Вызов функции. Передача аргументов функции. Возврат значений из функции. Локальные переменные функции: чем они отличаются от обычных переменных?

n. Массивы

Базовые операции с массивами: получение по индексу, изменение элемента, добавление элемента в конец массива, проход по массиву в цикле.

o. Чтение из стандартного входа или из файла

Команда `read` и её опции `-p`, `-r`, `-a`, `-s`. Использование `read` вместе с `while` для чтения строк из файла. Команда `readarray`.

8. Процессы

a. Основные понятия, связанные с процессами

Программа. Процесс. Родительский и дочерний процесс. Системные вызовы `fork` и `exec`. Идентификатор процесса. PID. PPID. Контекст процесса. Переключение контекста. Потоки. Разница между процессами и потоками. Квант процессорного времени. Планировщик.

b. Просмотр информации о процессах

Команда `ps` и её опции: `aux`, `-e`, `-f`, `-u`, `-p`, `-C`, `--ppid`, `--sort`, `-o`. Поля в выводе команды `ps`:

- | | |
|---------------|---------|
| • pid | • stat |
| • uid / user | • %cpu |
| • gid / group | • %mem |
| • comm | • rss |
| • cmd | • vsz |
| • args | • etime |

Команда `pstree`.

c. Идентификаторы процессов

PID. PPID. Как узнать идентификатор процесса? Как узнать идентификатор родительского процесса? Как узнать идентификатор дочерних процессов? Переменные `$$` и `#!`. Переменная `$PPID`.

d. Сигнал

Что такое сигнал? Номер сигнала. Основные сигналы:

- | | |
|-----------|-----------|
| • SIGTERM | • SIGSTOP |
| • SIGHUP | • SIGTSTP |
| • SIGINT | • SIGCONT |
| • SIGKILL | • SIGCHLD |

Команда `kill`. Как послать определённый сигнал процессу? Команды `killall` и `pkill`.

e. Команда `trap`

Перехват сигналов с помощью `trap`. Игнорирование сигналов. Какие сигналы нельзя перехватить и игнорировать? Обработка выхода из скрипта (`EXIT`). Обработка ошибок (`ERR`).

f. Группы процессов и сессии

Группа процессов. PGID. Какие процессы входят в группу процессов? Как послать сигнал всем процессам в группе? Лидер группы. Сессия. SID. Лидер сессии.

g. Состояния процессов

- Running / Runnable
- Sleeping
- Uninterruptible sleep
- Stopped
- Zombie

В каких случаях процесс попадает в то или иное состояние? Процессы, находящиеся в каких состояниях, нельзя завершить, посыл им сигналы, в том числе сигнал `SIGKILL`?

h. Управление заданиями

Фоновый и передний режим. Как запустить процесс в фоновом режиме? Команды `jobs`, `fg`, `bg`. Использование `kill` для послыки сигналов процессам из `jobs`. Как остановить и возобновить процесс? Какой сигнал и каким процессам посылаются при следующих действиях:

- Нажатии `Ctrl-C` в терминале
- Нажатии `Ctrl-Z` в терминале
- Закрытии окна терминала

i. Отсоединённый процесс

Отсоединение процесса. Команда `nohup`. Команда `disown`. Команда `setsid`. Демоны. Как написать скрипт, при запуске которого создавался бы процесс-демон?

j. Приоритет процессов

Приоритет и `niceness`. Какие значения может принимать `niceness`? Команда `nice` и её опция `-n`. Команда `renice` и её опции `-n` и `-p`.

k. **Мониторинг процессов, программа top**

Команда `top`. Какую информацию она показывает? Что означают значения `load average`?

Что означают поля, показывающие распределение времени процессора (%CPU(s)):

- | | | |
|------|------|------|
| • us | • ni | • wa |
| • sy | • id | • hi |

Что означают поля:

- | | | |
|--------|--------|-----------|
| • PID | • VIRT | • %CPU |
| • USER | • RES | • %MEM |
| • PR | • SHR | • TIME+ |
| • NI | • S | • COMMAND |

Программа `htop`.

l. **Директория /proc**

Директория `/proc/<PID>` и файлы в этой директории:

- | | |
|-----------|----------|
| • cmdline | • exe |
| • environ | • fd/1 |
| • maps | • status |
| • cwd | • stat |

9. **systemd**

a. **Основы**

Что такое `systemd` и какую роль он выполняет в Linux? Юнит в `systemd`. Типы юнитов:

- service
- target
- timer
- path

b. **Команда systemctl**

Команда `systemctl` и её подкоманды:

- | | |
|---------------------|---------------------------|
| • systemctl start | • systemctl status |
| • systemctl stop | • systemctl cat |
| • systemctl restart | • systemctl reload |
| • systemctl enable | • systemctl daemon-reload |
| • systemctl disable | • systemctl list-units |

c. **Юнит-файлы**

В каких директориях `systemd` ищет юнит-файлы? Как создать свой сервис? Строение юнит-файла.

Секции. Директивы секции `[Unit]`:

- Description
- After/Before
- Wants/Requires
- Condition*

Директивы секции `[Service]`:

- | | |
|--|----------------------------------|
| • Type, типы simple, forking, oneshot, notify. | • Environment/EnvironmentFile |
| • ExecStart | • PIDFile |
| • ExecStop | • TimeoutStartSec/TimeoutStopSec |
| • Restart | • KillMode |
| • User/Group | • RemainAfterExit |

Директива секции `[Install]`:

- WantedBy

Таргеты по умолчанию:

- `rescue.target`
- `multi-user.target`
- `graphical.target`
- `default.target`

d. **Таймеры**

Зачем нужны timer-юниты? Как создать timer-юнит, который бы запускал сервис по расписанию?

Директивы:

- | | |
|----------------------------|-----------------------------------|
| • <code>Unit</code> | • <code>Persistent</code> |
| • <code>OnBootSec</code> | • <code>RandomizedDelaySec</code> |
| • <code>OnActiveSec</code> | • <code>OnCalendar</code> |

e. **Path-юниты**

Зачем нужны path-юниты? Как создать path-юнит, который бы запускал сервис при изменении файла?

Директивы:

- `Unit`
- `PathExists`
- `PathChanged`

f. **Журнал**

Команда `journalctl`.

Задача 1. Что добавить

- Директория `.git`
- Как можно полностью удалить коммиты из git?
- `git commit --amend`
- `git commit --fixup`
- `git diff --name-only`
- Удаление ветки на удалённом сервере `git push --delete`
- `--rebase-merges`
- `git check-ignore`
- `locate`, `updatedb`
- `xxd`, `hexdump`
- `pgrep`
- `free`
- Какой диапазон значений принимает код возврата