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`
   - `echo` и её опции `-n` и `-e`
   - `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) **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 просмотр 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 $\sim$ и $\wedge$.

(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` и чем опасно это состояние?

(i) **Слияние**
Слияние веток. Команда `git merge`. Слияние перемоткой (fast-forward merge) и 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` и `git rebase --skip`.

- `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 потери 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:

- `pick`
- `reword`
- `edit`
- `squash`
- `fixup`
- `drop`

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 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 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. **Команды, которые часто используются в конвейерах**

- `tac`
- `head`
- `tail`
- `wc`

b. **Потоки и перенаправление**
Потоки `stdin`, `stdout` и `stderr`. Перенаправление потоков.

c. **Синтаксис подстановки команд**

d. **Конвейеры**

e. **Wildcards**

f. **Команда `find`**

g. **Команда `xargs`**

h. **Команда `grep`**

i. **Команда `tee`**

5. **Пользователи и права доступа Linux**

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

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

В каком формате хранится информация в каждом из файлов? Как хранится информация о паролях пользователей?

b. **Просмотр информации о пользователях и группах**
Команды `id`, `groups` и `getent`.

c. **Работа с пользователями**

- `useradd`
- `usermod`
- `userdel`

Блокировка пользователей.

d. **Работа с паролями**
Команда `passwd`.

e. `sudo` **и** `su`

f. `Права доступа`
Просмотр прав доступа с помощью `ls`. Символьное и числовое представление прав доступа. Конвертация из одного представления в другое. Изменение прав доступа с помощью команды `chmod`. На что влияют права доступа обычных файлов? На что влияют права доступа директорий?

g. **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 информации**

b. **Просмотр информации о дисках, разделах и точках монтирования**

- `lsblk`
- `blkid`
- `findmnt`

c. **Просмотр информации об использованной памяти**

- `df`
- `du`

d. **Таблица разделов**
MBR и GPT

e. **Команда** `dd`

f. **Ссылки**
Мягкие и жёсткие.

g. **Файловые системы**
Блоки. Суперблок. Таблица inode. Что хранится в inode-ах?

h. **Распространённые файловые системы**
ext4, FAT, NTFS.

7. **Язык Bash**

a. **Переменные Bash**
Создание и использование переменных в bash. Переменные среды. Команды `unset` и `export`. Файл `.bashrc`.

b. **Переменная среды** `PATH`

c. **Аргументы**
Передача аргументов в Bash.

d. **Коды возврата**

e. **Управляющая конструкция** `if fi`

f. **Управляющая конструкция** `case esac`

g. **Циклы**

h. **Манипуляции со строками в bash**

i. **Работа с целыми числами**

j. **Функции**

k. **Массивы и словари**

l. **Чтение из стандартного входа или из файла**

8. **Процессы**

   a. **Основные определения**
   Что такое программа? Что такое процесс?

   b. **Управление заданиями**
   Фоновый и передний режим. Как запустить процесс в фоновом режиме? Команды `jobs`, `fg`, `bg`. Использование `kill` для того, чтобы посылать сигналы процессам из `jobs`. Как остановить и возобновить процесс?

   c. **Просмотр информации о процессах**
   Команда `ps` и её опции `-e`, `-f`, `aux`, `-u`, `-o`. Команда `pstree`.

   d. **Идентификаторы**
   Идентификатор процесса PID. Как узнать идентификатор процесса? Как узнать идентификатор дочернего процесса? Группа процессов. Сессия.

   e. **Сигнал**
   Что такое сигнал? Основные сигналы:

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

   Как послать сигнал процессу?

   f. **Перехват сигналов**
   Команда `trap`. Перехват сигналов. Игнорирование сигналов.

   g. **Отсоединённый процесс**
   Отсоединение процесса. Команда `nohup`.

   h. **Состояния процессов**

   - Running
   - Sleeping
   - Uninterruptible sleep
   - Stopped
   - Zombie

   В каких случаях процесс попадает в то или иное состояние?

   i. **Мониторинг процессов**
   Команда `top` и `htop`. Директория `/proc`.

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`**
Подкоманды:

- `systemctl start`
- `systemctl stop`
- `systemctl enable`
- `systemctl disable`
- `systemctl status`
- `systemctl cat`
- `systemctl reload`
- `systemctl daemon-reload`
- `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`.
- `ExecStart`
- `ExecStop`
- `Restart`
- `User`/`Group`
- `Environment`/`EnvironmentFile`
- `PIDFile`
- `TimeoutStartSec`
- `KillMode`

Директива секции `[Install]`:

- `WantedBy`

Таргеты по умолчанию:

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