t.me/linuxkalii

Шпаргалка по командам Linux для среднего и продвинутого уровня

1. Повседневные задачи (навигация, работа с файлами, ввод/вывод)

- `pwd` – показывает полный путь текущего рабочего каталога (print working directory). *Пример:* `pwd` выведет, например, `/home/user/projects` – путь к текущей папке.
- `ls` – отображает содержимое каталога (список файлов и папок). Часто используется с флагами `-lah` для показа скрытых файлов (начинающихся с `.`) и подробной информации (размер, права, дата) [1] . *Пример:* `ls -lah /var/log` покажет список всех файлов в `/var/log` с деталями.
- `cd` – смена каталога (change directory). Используйте `cd /полный/путь` для перехода по абсолютному пути или `cd имя_папки` для перехода по относительному пути. `cd ..` поднимается на уровень вверх, `cd -` возвращается в предыдущую директорию [2] , `cd ~` или просто `cd` переходит в домашний каталог пользователя.
- `mkdir` – создание новых директорий. *Пример:* `mkdir new_dir` создаст папку *new_dir*. Флаг `-p` создаёт вложенные каталоги, например `mkdir -p dir/subdir` создаст одновременно *dir* и *dir/subdir* при их отсутствии [3] .
- `cp` – копирование файлов или папок (copy). *Пример:* `cp file.txt /tmp/` скопирует *file.txt* в каталог */tmp/*. Флаг `-r` нужен для рекурсивного копирования директорий. Флаг `-i` спросит подтверждение при перезаписи существующего файла [4] .
- `mv` – перемещение или переименование (move). *Пример:* `mv old_name.txt new_name.txt` переименует файл. `mv /path/dir/* ./` переместит все файлы из */path/dir/* в текущий каталог [5] . С флагом `-i` команда запросит подтверждение перед перезаписью файлов при перемещении [6] .
- `rm` – удаление файлов или папок (remove). *Пример:* `rm file.txt` удалит файл. Для удаления папки и её содержимого используется рекурсивный флаг: `rm -r folder` (осторожно: файлы удаляются безвозвратно) [7] . Флаг `-f` принудительно удалит без запроса, а `-i` – с запросом подтверждения.
- `touch` – создание пустого файла или обновление времени изменения файла. *Пример:* `touch example.txt` создаст файл *example.txt* (или обновит *timestamp*, если файл уже существует).
- `cat` – выводит содержимое файла или объединяет несколько файлов (concatenate). *Пример:* `cat /etc/os-release` выведет текст содержимого файла.
- `head` / `tail` – показывают начало или конец вывода/файла. По умолчанию 10 строк. *Пример:* `head -n 5 large.log` выведет первые 5 строк файла, а `tail -f large.log` будет **постоянно следить** за появлением новых строк в конце лога.
- **Переадресация ввода/вывода:** оператор `>` перезаписывает вывод команды в файл, `>>` дописывает в конец файла, `<` берёт ввод из файла, а `|` (pipe) перенаправляет вывод одной команды на ввод другой. *Пример:* `ls /etc | grep conf > conf_files.txt` — результат `ls` фильтруется по слову "conf" и сохраняется в файл.

- `less` – постраничный просмотр текста (пейджер). Позволяет прокручивать длинный вывод вверх/вниз (клавишами `PgUp/PgDn` или стрелками). *Пример:* `ls -l /bin | less` — листать длинный список файлов. (Нажмите `Q` для выхода из less.)

2. Работа с сетью (проверка подключения, конфигурация, порты)

- `ping` – проверяет доступность узла в сети, отправляя ICMP эхо-запросы и измеряя время отклика [8]. *Пример:* `ping -c 4 8.8.8.8` посылает 4 пакета на адрес *8.8.8.8* и покажет задержку и процент потерь.
- `ifconfig` / `ip` – вывод и настройка сетевых интерфейсов. `ifconfig` показывает IP-адреса и состояние интерфейсов (устаревшая утилита, в новых системах заменена на `ip`). *Пример:* `ip addr show` (или коротко `ip a`) отобразит адреса всех интерфейсов, `ip route show` – таблицу маршрутизации.
- `netstat` / `ss` – отображает сетевые соединения и порты. `netstat -tulnp` перечисляет прослушиваемые TCP/UDP порты и соответствующие PID процессов (может требовать установку **net-tools**). Современный аналог: `ss -tulwnp` (работает быстрее и установлен по умолчанию) [9].
- `nmap` – сетевой сканер портов и служб (Network Mapper). *Пример:* `nmap -p 1-1000 example.com` просканирует 1000 портов на хосте *example.com* и покажет какие открыты.
- `curl` – отправляет запросы к URL (получает данные по HTTP/HTTPS, FTP и т.д.). *Пример:* `curl -I https://example.com` выведет заголовки ответа HTTP от *example.com*. Используйте `-O` для сохранения загруженного файла или `-d` для отправки POST-данных.
- `wget` – скачивает файлы по URL (в консоли). *Пример:* `wget https://wordpress.org/latest.zip` скачает файл *latest.zip* в текущий каталог.
- `ssh` – подключение к удалённому серверу по протоколу SSH (Secure Shell). *Пример:* `ssh user@host` удалённо зайдёт на *host* под пользователем *user*. Используйте ключ `-p` для нестандартного порта, а `-L` или `-R` для проброса портов (порт форвардинг).
- `scp` – копирование файлов по SSH между хостами (Secure Copy). *Пример:* `scp backup.tar user@remote:/backup/` скопирует файл *backup.tar* на удалённый сервер *remote* в каталог */backup/*.
- `dig` / `nslookup` – утилиты для DNS-запросов. *Пример:* `dig example.com ANY` покажет DNS-записи домена *example.com*. `nslookup example.com` – простой запрос DNS A-записи.
- `traceroute` – трассировка маршрута до узла (список промежуточных узлов). *Пример:* `traceroute 8.8.8.8` покажет через какие узлы и с какими задержками проходят пакеты до IP *8.8.8.8*.
- `telnet` / `nc` – утилиты для тестирования подключения к портам. *Пример:* `telnet example.com 25` попытается подключиться к *example.com* на порт 25 (SMTP) и покажет, удалось ли соединиться. Утилита `nc` (netcat) более современна: `nc -vz example.com 25` проверит доступность порта с выводом результата.

3. Управление файлами и файловой системой (копирование, поиск, монтирование)

- `find` – поиск файлов и каталогов по заданным критериям. Команда рекурсивно обходит указанную директорию. *Пример:* `find . -name "*.txt"` найдет все файлы с расширением `.txt` в текущем каталоге и подкаталогах [10]. Можно комбинировать условия: `find /var/log -type f -size +1M` – найти файлы >1 МБ в */var/log*. Флаг `-exec cmd {} \;` позволяет выполнить команду для каждого найденного результата.

- `grep` – поиск совпадений в тексте файлов или выводе команд (global regular expression print). Ищет строки, совпадающие с шаблоном (поддерживаются регулярные выражения). *Пример:* `grep -R "ERROR" /var/log` рекурсивно найдет строки с “ERROR” в файлах каталога */var/log* [11]. Флаг `-i` отключает чувствительность к регистру, `-n` выводит номера строк, а `-v` инвертирует поиск (показывает строки без совпадений).
- `locate` – мгновенный поиск файлов по имени через предварительно проиндексированную базу. *Пример:* `locate *.pdf` быстро найдет все файлы с расширением .pdf в системе. *(Требуется установленный* `updatedb` *сервис для индексации.)*
- `tar` – архивация/распаковка файлов (Tape ARchiver). *Примеры:* `tar -czf archive.tar.gz /path/to/dir` создаст сжатый gzip-архив *archive.tar.gz* из содержимого папки */path/to/dir*. `tar -xvf archive.tar.gz` распакует архив в текущий каталог. Основные флаги: `-c` создать, `-x` извлечь, `-z` сжатие gzip, `-v` подробный вывод, `-f <файл>` указать имя архива.
- `zip` / `unzip` – альтернативно, упаковка в zip-архив и распаковка. *Пример:* `zip -r files.zip dir1 dir2` заархивирует *dir1* и *dir2* в *files.zip*. `unzip files.zip` извлечет содержимое.
- `df` – показывает свободное/занятое место на файловых системах (disk free). *Пример:* `df -h` отобразит сводку по дискам в удобном для чтения формате (с К/М/ГБ) [12].
- `du` – оценивает используемое пространство в каталогах или файлах (disk usage). *Пример:* `du -sh /var/log` покажет суммарный размер содержимого каталога */var/log* (флаг `-h` делает размеры удобочитаемыми, `-s` для итога без детализации) [13] [14].
- `mount` – монтирование файловой системы (подключение раздела, диска или образа к каталогу). Используется для подключения устройств (USB, диски) или сетевых ресурсов к общей файловой системе Linux [15]. *Пример:* `sudo mount /dev/sdb1 /mnt/usb` примонтирует устройство */dev/sdb1* в каталог */mnt/usb*. После работы отмонтируйте с помощью `umount /mnt/usb` (отключение диска).
- `lsblk` – отображает список блочных устройств (диски, разделы) в виде дерева. Полезно перед монтированием, чтобы узнать правильные имена устройств. *Пример:* `lsblk -f` покажет устройства, их точки монтирования и файловые системы.
- `fsck` – проверка и исправление файловой системы (file system check). *Пример:* `sudo fsck -y /dev/sda2` проверит раздел */dev/sda2* на ошибки и автоматически попытается их исправить (`-y` отвечает "yes" на запросы). **Внимание:** запускать на отмонтированных или в режиме восстановления, иначе может повредить смонтированную ФС.
- `ln` – создание ссылок на файлы. `ln file link` создаст жёсткую ссылку (работает как второй именованный путь к тому же файлу), `ln -s target link` – символическую (soft link), указывающую на *target* (может быть каталог или файл). *Пример:* `ln -s /path/original.txt shortcut.txt` создаст ярлык *shortcut.txt* на *original.txt*.

4. Процессы и ресурсы (мониторинг, системные службы)

- `ps` – snapshot списка процессов. Отображает запущенные в системе процессы (не интерактивно). *Примеры:* `ps aux` выводит все процессы всех пользователей с подробностями (UID, PID, загрузка, командная строка) [16]. Для фильтрации по имени можно использовать конвейер: `ps aux | grep nginx` ищет процессы, связанные с *nginx*.
- `top` – динамический монитор процессов в реальном времени (обновляет список каждые несколько секунд). Показывает активные процессы, загрузку CPU, использование памяти и др. Колонка PID – идентификатор процесса, USER – владелец процесса и т.д. [17]. *Совет:*

Внутри `top` нажмите **H** для справки по горячим клавишам (например, **P** сортирует по CPU, **M** по памяти, **Q** – выход).

- `htop` – улучшенная версия top (более наглядный интерактивный монитор). Отображает процессы цветными блоками, графики загрузки CPU/памяти, позволяет прокручивать список, искать процессы. Требует отдельной установки на многих системах. Полезна для глубокой диагностики нагрузки [18].
- `free` – показывает объём свободной и используемой памяти (RAM и swap). *Пример:* `free -h` выведет статистику памяти в удобном формате (в гига/мегабайтах). Колонки: total, used, free, available и т.д.
- `uptime` – выводит время работы системы (со времени последней перезагрузки) и среднюю нагрузку (load average). *Пример:* `uptime` может показать, например: *"14:32:16 up 5 days, 3:21, 2 users, load average: 0.15, 0.10, 0.09"*.
- `kill` – посылает сигнал указанному процессу по PID. Обычно используется для корректного завершения или принудительного прерывания процесса. *Пример:* `kill 1234` посылает процессу с PID 1234 сигнал SIGTERM (по умолчанию) для завершения [19]. Если процесс не завершается, `kill -9 1234` посылает SIGKILL (жёсткое прерывание). **NB:** Перед `-9` попробуйте мягкие сигналы или `kill -15` (SIGTERM).
- `pkill` / `killall` – завершение процессов по имени. `pkill name` отправит сигнал всем процессам, чьё имя соответствует *name* (можно добавлять `-u user` для процессов конкретного пользователя). `killall program` делает то же самое. *Пример:* `pkill -HUP nginx` перезапустит все процессы *nginx* послав им сигнал HUP.
- `nice` / `renice` – изменяют приоритет процесса. `nice -n 5 command` запустит *command* с пониженным приоритетом (+5). `renice -n -10 -p 1234` повысит приоритет процесса с PID 1234 (более высокий приоритет = более негативное значение nice).
- `lsof` – список открытых файлов (и сокетов) процессов (list open files). *Пример:* `lsof -i :80` покажет процессы, слушающие порт 80, и связанные с ними сетевые соединения.
- `systemctl` – основной инструмент управления службами (демонами) в системах с systemd. Позволяет запускать, останавливать, перезапускать службы и проверять их статус [20]. *Примеры:* `sudo systemctl status apache2` – показывает статус службы *apache2* (работает ли, журнал последних сообщений) [20]. `sudo systemctl stop apache2` / `start apache2` – остановить или запустить службу; `restart` – перезапустить ее [21]. `enable` / `disable` – включить или отключить автозапуск службы при старте системы.
- `journalctl` – просмотр логов systemd (журнала). *Пример:* `journalctl -u nginx -f` покажет сообщения журнала для службы *nginx* и будет следить за новыми (флаг `-f`). Используйте `-b` для логов текущей загрузки, `-k` для dmesg (сообщения ядра), или комбинации флагов для фильтрации по времени, приоритету и т.д.
- `systemd-analyze` – инструмент для диагностики процесса загрузки systemd. *Пример:* `systemd-analyze blame` покажет, сколько времени занял старт каждой службы при последней загрузке; `systemd-analyze critical-chain` – последовательность задержек. Полезно для оптимизации старта системы.

5. Безопасность и разрешения (права доступа, sudo, брандмауэр)

- `chmod` – изменение прав доступа (change mode). Права в Linux выражаются для **владельца/группы/остальных** в виде комбинации *чтение (r), запись (w), выполнение (x)*. *Примеры:* `chmod 600 file` установит права *rw-------* (только владелец может читать/писать) для *file*. `chmod -R u+rX,g-rwx,o-rwx /srv` рекурсивно выдаст владельцу право чтения и выполнения на файлы/папки в */srv*, убрав все права для группы и остальных.

- `chown` – смена владельца файла/директории. Синтаксис: `chown <новый_владелец>:<новая_группа> файл`. *Пример:* `sudo chown user:users notes.txt` сделает *user* владельцем файла *notes.txt* и назначит группу *users*. Флаг `-R` выполняет операцию рекурсивно для папок.
- `chmod` **vs** `chown` **заметка:** первая меняет **права**, вторая — **владельца**. Проверить текущие права и владельцев можно командой `ls -l`.
- `sudo` – выполнение команды с правами суперпользователя (Administrator/root). Требует ввода пароля текущего пользователя (если он в sudoers). *Пример:* `sudo nano /etc/hosts` откроет файл *hosts* от имени администратора (т.е. позволит сохранить изменения) [22]. Используйте `sudo -i` или `sudo su` для получения root-оболочки (но лучше по возможности выполнять единичные команды через sudo).
- `su` – смена пользователя (switch user). Вызов `su` без аргументов переключается на root-пользователя (нужно ввести root-пароль). `su username` – войти под указанным пользователем (требуется знать его пароль). После `su` окружение меняется на новое (в т.ч. домашняя директория). Выйти из сессии su – командой `exit` или сочетанием **Ctrl+D**.
- `passwd` – смена пароля пользователя. *Пример:* `passwd` (без параметров) запросит ввод нового пароля для текущего пользователя. Администратор может менять пароли других: `sudo passwd username` задаст новый пароль для *username*. Команда также используется для блокировки учётной записи: `passwd -l username` (lock) и разблокировки `passwd -u username` (unlock).
- `umask` – устанавливает маску создания файлов (какие права **не** будут выставлены по умолчанию при создании новых файлов/папок). *Пример:* `umask 077` – новые файлы будут доступны только владельцу (никаких прав для группы и других). Текущую маску можно посмотреть просто командой `umask`.
- `ufw` – простой встроенный фаервол (Ubuntu). *Примеры:* `sudo ufw enable` активирует фаервол, `sudo ufw allow 22/tcp` откроет порт 22 (SSH), `sudo ufw status` покажет текущие правила. UFW упрощает работу с iptables (внутри себя применяет правила **iptables**).
- `iptables` – утилита для настройки брандмауэра на низком уровне (в ядре Linux, таблицы сетевых фильтров). Очень мощная, но требующая понимания. *Пример:* `sudo iptables -L -n -v` покажет текущие правила фильтрации. Создание правила: `sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT` разрешит входящие соединения на 80 порт.
- `fail2ban` – инструмент для защиты сервисов от перебора паролей и брутфорс-атак. Мониторит логи выбранных сервисов (SSH, FTP, веб и др.) и при большом числе неудачных попыток или подозрительной активности банит IP-нарушителя через правила фаервола на заданное время [23]. *Пример:* после установки, выполните `sudo fail2ban-client status` чтобы увидеть активные «тюрьмы» (jails) и количество заблокированных IP. `sudo fail2ban-client status sshd` покажет статистику по jail для SSH [24]. Параметры фильтров настраиваются в */etc/fail2ban/jail.local* (количество попыток, время бана и т.д.).

6. Сценарии и автоматизация (shell-скрипты, планировщики задач)

- **Bash-скрипты** – автоматизация повторяющихся действий через написание сценариев командного интерпретатора (bash). Скрипт – текстовый файл с командами. Начинается с shebang (`#!/bin/bash`). Чтобы запустить, нужно дать файлу права на исполнение (`chmod +x script.sh`) и выполнить `./script.sh` (либо `bash script.sh`). *Пример скрипта:*

```
#!/bin/bash
echo "Привет, $USER! Сегодня $(date)."
```

Этот скрипт при запуске поприветствует текущего пользователя и выведет текущую дату.

- `cron` – планировщик заданий, позволяющий запускать команды по расписанию. Отредактировать кронтабы можно через `crontab -e` (для текущего пользователя). Формат записи: `минуты часы день_месяца месяц день_недели команда`. *Пример:* `0 3 * * * /home/user/backup.sh` – выполнить */home/user/backup.sh* каждый день в 3:00 ночи. Команда `crontab -l` покажет текущее расписание. **Совет:** использовать `crontab -e` под пользователем *root* для системных задач и под обычным пользователем для пользовательских задач.
- `at` – единоразовый запуск задачи в заданное время (альтернатива cron для однократных задач). *Пример:* `echo "notify-send 'Done!'" | at 14:00` поставит команду на выполнение в 14:00 сегодняшнего дня. Команда `atq` покажет очередь заданий, `atrm` удалит задание.
- **Systemd timers** – альтернатива cron на системах с systemd. Позволяют запускать службы или скрипты по расписанию через .timer–юниты. *Пример:* создание пары файлов *myjob.service* (описывает задачу) и *myjob.timer* (описывает расписание) в */etc/systemd/system/*. В файле *.timer* задаётся OnCalendar (например, `OnCalendar=*-*-* 3:00:00` для ежедневного запуска в 3:00). После этого `sudo systemctl enable --now myjob.timer` активирует таймер. Проверить активные таймеры можно командой `systemctl list-timers`.
- `watch` – периодически выполняет заданную команду и обновляет вывод. Удобно для мониторинга. *Пример:* `watch -n 5 'df -h /home'` будет каждые 5 секунд показывать свободное место на */home*. По умолчанию обновляет каждые 2 секунды; клавиша **Ctrl+C** останавливает.
- **Автоматизация через скрипты:** Используйте условные операторы (`if ... fi`), циклы (`for` , `while`) и конструкции &&/|| для более сложной логики. Например, `command1 && command2` выполнит *command2* **только если** *command1* завершилась успешно, а `command1 || command2` – выполнит *command2* если *command1* завершилась с ошибкой. Комбинируя это с cron или systemd timers, можно добиться гибкого планирования задач.

7. Управление пакетами (установка и обновление ПО)

(Примечание: пакетные менеджеры различаются в зависимости от дистрибутива.) - `apt` **(Debian/Ubuntu)** – менеджер пакетов для Debian-подобных систем. **Основные команды:** `sudo apt update` (обновить индекс пакетов), `sudo apt upgrade` (обновить установленные пакеты), `sudo apt install пакет` (установить пакет) [25] , `sudo apt remove пакет` (удалить пакет), `sudo apt search имя` (поиск пакета по имени), `sudo apt autoremove` (удалить неиспользуемые зависимости). *Пример:* `sudo apt install nginx` установит веб-сервер *nginx* (потребуются права sudo). - `dnf` **(Fedora/Red Hat)** – менеджер пакетов для RPM-систем (замена yum). Команды аналогичны: `sudo dnf install пакет` , `sudo dnf upgrade` (обновить все пакеты), `sudo dnf search имя` , `sudo dnf remove пакет` . *Пример:* `sudo dnf install httpd` установит Apache HTTP Server на Fedora/RHEL. - `pacman` **(Arch Linux)** – менеджер пакетов для Arch/Manjaro. **Основные опции:** `-S` (установка пакета из репозиториев), `-R` (удаление пакета), `-Sy` (обновить индекс), `-Su` (обновить установленные пакеты), часто комбинируется в `-Syu` (полное обновление системы). *Примеры:* `sudo pacman -Syu` обновит систему, `sudo pacman -S package_name` установит пакет, `pacman -Qs name` ищет установленный пакет по имени, `sudo pacman -Rsn package` удалит пакет и неиспользуемые зависимости. - `snap`

(Универсальные пакеты) – менеджер пакетов snap (sandboxed приложения). *Примеры:* `snap find ключевое_слово` – поиск snap-пакета, `sudo snap install name` – установка, `snap list` – список установленных snap-пакетов, `sudo snap remove name` – удаление пакета. - `flatpak` **(Универсальные пакеты)** – система распространения приложений Flatpak. *Примеры:* `flatpak search имя` – поиск приложения, `flatpak install remote name` – установка (например, из репозитория *flathub*: `flatpak install flathub com.spotify.Client`), `flatpak list` – список установленных, `flatpak update` – обновление, `flatpak remove name` – удаление приложения. - `dpkg` **/** `rpm` – низкоуровневые инструменты установки локальных пакетов. `dpkg -i file.deb` установит .deb-пакет (Debian), `rpm -ivh file.rpm` установит .rpm-пакет (RedHat/Fedora). Обычно используются, если вы скачали пакет вручную. Удаление: `dpkg -r package_name` или `rpm -e package_name`. - `pip` – менеджер пакетов Python (для установки библиотек Python). *Пример:* `pip install requests` установит пакет *requests* для Python. Добавьте `--user` для установки в домашний каталог пользователя (без sudo). **Примечание:** для Python3 может использоваться команда `pip3`.

8. Лайфхаки, горячие клавиши и полезные алиасы

- **Горячие клавиши Bash:** Использование сочетаний клавиш ускоряет работу в терминале [26]. Например: **Ctrl+C** – прервать текущую выполняющуюся команду; **Ctrl+Z** – приостановить (suspend) выполнение команды (отправить в фон); **Ctrl+D** – обозначить конец ввода (EOF) или выйти из терминала, если ввод не ожидался; **Tab** – автодополнение имен файлов/команд; **Ctrl+A** / **Ctrl+E** – перейти в начало/конец текущей команды; **Alt+B** / **Alt+F** – переместиться назад/вперёд на одно слово; **Ctrl+U** / **Ctrl+K** – удалить всё от курсора до начала/конца строки; **Ctrl+R** – поиск по истории команд (начните набирать фрагмент команды); **Ctrl+L** – очистить экран (аналог `clear`, при этом текущая строка останется неизменной) [26].
- **История команд:** Ваша командная строка хранит историю вводимых команд. Используйте `history` для вывода списка последних команд с номерами. Можно повторно вызвать команду по номеру: `!<номер>` или воспользоваться удобными шаблонами: `!!` – повтор последней команды (полностью) с теми же аргументами [27]; `sudo !!` – повтор последней команды, но с sudo (очень полезно, когда забыли запустить команду от администратора) [27]; `!$` – подставляет последний аргумент предыдущей команды (например, после ввода `mkdir /long/path/to/dir` команда `cd !$` быстро перейдет в */long/path/to/dir*); `^старое^новое` – быстрое исправление опечатки: выполнит предыдущую команду, заменив первое вхождение слова *старое* на *новое*.
- **Алиасы (alias):** позволяют создавать сокращения для часто используемых команд [28]. *Пример:* `alias ll='ls -lhA'` создает команду **ll** как короткий аналог `ls -lhA` (подробный листинг со скрытыми файлами). После добавления алиаса в терминале он будет действовать до конца сессии; чтобы алиас всегда был доступен, добавьте строку в ваш *~/.bashrc*. Посмотреть все текущие алиасы: `alias` без аргументов [28]. Удалить алиас: `unalias имя` [29].
- **Полезные алиасы и функции:** Настройте свое окружение под себя. Например:
- `alias grep='grep --color=auto'` – подсветка найденных совпадений в grep.
- `alias ..='cd ..'` и `alias ...='cd ../..'` – быстрый подъем по каталогам.
- `alias gs='git status'`, `alias ga='git add'` – сокращения для часто используемых git-команд (для разработчиков).
- Функция в bash (в *~/.bashrc*):

```
extract() { case "$1" in *.tar.gz) tar -xzf "$1" ;; *.zip) unzip
"$1" ;; esac }
```

Такая функция **extract** позволит одной командой распаковать .zip или .tar.gz файл в текущем каталоге.

- **Другие лайфхаки:**
- Используйте `screen` или `tmux` для запуска длительных процессов в разделяемых сессиях терминала (можно отключиться и подключиться позже, процесс не прервется).
- Команда `!!:gs/OLD/NEW` заменит во **всех** аргументах предыдущей команды слово OLD на NEW и выполнит команду (g - глобально, s - substitute).
- Быстро создать последовательность директорий/файлов: `mkdir folder_{1..5}` создаст *folder_1* ... *folder_5*. Аналогично, `touch file{A..C}.txt` создаст *fileA.txt*, *fileB.txt*, *fileC.txt*.
- Запустить простой веб-сервер в текущем каталоге: `python3 -m http.server 8000` – поднимет HTTP-сервер на порту 8000, доступный для просмотра содержимого каталога через браузер.
- Будьте любознательны: многие команды имеют полезные опции. Используйте `--help` или читайте мануалы (`man команда`) для изучения возможностей. **Happy hacking!**

Ссылки: Команды и описания основаны на официальных руководствах и популярных справочниках [1] [20] [23] [27] и предназначены для быстрого ознакомления. Пользуйтесь ими, чтобы работать в Linux быстрее и эффективнее!

---

[1] [2] [3] [4] [5] [6] [7] [13] [14] [16] [19] Топ необходимых Linux-команд для разработчика [Шпаргалка] / Хабр
https://habr.com/ru/articles/788970/

[8] Сетевые команды — ALT Linux Wiki
https://www.altlinux.org/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D1%8B%D0%B5_%D0%BA%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D1%8B

[9] 4 способа узнать, какие порты прослушиваются в Linux (открыты)
https://blog.sedicomm.com/2019/02/27/4-sposoba-uznat-kakie-porty-proslushivayutsya-v-linux/

[10] [11] [12] [15] [20] [21] [28] [29] 20 команд для пользователей среднего уровня в Linux
https://blog.sedicomm.com/2023/06/09/20-komand-dlya-polzovatelej-srednego-urovnya-v-linux/

[17] [18] [22] [25] Основные команды Linux терминала с примерами использования
https://selectel.ru/blog/basic-linux-commands/

[23] Руководство по пентесту и защите от киберугроз на Linux и Kali Linux.pdf
file://file-XFQcMZ3mZWwKGvvY1frRkT

[24] Настройка Fail2ban для защиты SSH-соединения | Джино • Справка
https://jino.ru/spravka/articles/f2b.html

[26] Гайд по командам Linux-терминала с примерами: основные возможности командной строки / Skillbox Media
https://skillbox.ru/media/code/osnovnye-komandy-i-goryachie-klavishi-terminala-linux-a-takzhe-unix-macos-i-freebsd/

[27] Как использовать команду sudo в Linux
https://wiki.merionet.ru/articles/kak-ispolzovat-komandu-sudo-v-linux