# tldr pages book

Simplified and community-driven man pages

*Generated on Wed Nov 5 04:58:02 2025*

Website: https://tldr.sh

GitHub: https://github.com/tldr-pages/tldr

# Android

# am

Менеджер активностей Android.

Больше информации: https://developer.android.com/tools/adb#am.

- Начать определённую активность:

  ```
  am start -n {{com.android.settings/.Settings}}
  ```

- Начать активность и передать в неё данные:

  ```
  am start -a {{android.intent.action.VIEW}} -d {{tel:123}}
  ```

- Начать активность соответствующую определенному действию и категории:

  ```
  am start -a {{android.intent.action.MAIN}} -c
  {{android.intent.category.HOME}}
  ```

- Преобразовать намерение в URI:

  ```
  am to-uri -a {{android.intent.action.VIEW}} -d {{tel:123}}
  ```

# bugreport

Показать отчет об ошибках Android.

Эту команду можно использовать только через **`adb shell`**.

Больше информации: https://cs.android.com/android/platform/superproject/+/main:frameworks/native/cmds/bugreport.

- Показать полный отчет об ошибках на устройстве Android:

  ```
  bugreport
  ```

# bugreportz

Создает заархивированный отчет об ошибках Android.

Эта команда может быть использована только через **`adb shell`**.

Больше информации: https://cs.android.com/android/platform/superproject/+/main:frameworks/native/cmds/bugreportz.

- Создать полный заархивированный отчет об ошибках устройства Android:

  `bugreportz`

- Показать прогресс выполняемой операции `bugreportz`:

  `bugreportz -p`

- Вывести содержимое отчета об ошибках Android в стандартный вывод:

  `bugreportz -s`

- Отобразить справку:

  `bugreportz -h`

- Отобразить версию:

  `bugreportz -v`

# cmd

Сервис менеджер Android.

Больше информации: https://cs.android.com/android/platform/superproject/+/main:frameworks/native/cmds/cmd/.

- Список всех запущенных сервисов:

  ```
  cmd -l
  ```

- Вызов конкретного сервиса:

  ```
  cmd {{alarm}}
  ```

- Вызов сервиса с аргументами:

  ```
  cmd {{vibrator}} {{vibrate 300}}
  ```

# dalvikvm

Виртуальная машина Android Java.

Больше информации: https://source.android.com/docs/core/runtime.

- Запустить Java-программу:

```
dalvikvm -classpath {{путь/к/файлу.jar}} {{classname}}
```

# dumpsys

Предоставляет информацию о системных службах Android.

Эту команду можно использовать только через **`adb shell`**.

Больше информации: https://developer.android.com/tools/dumpsys.

- Получить диагностическу для всех системных сервисов:

  ```
  dumpsys
  ```

- Получить диагностическу для конкретной системного сервисв:

  ```
  dumpsys {{сервис}}
  ```

- Список всех сервисов доступных через `dumpsys`:

  ```
  dumpsys -l
  ```

- Задать специфичные для сервиса аргументы:

  ```
  dumpsys {{сервис}} -h
  ```

- Исключить конкретный сервис из диагностики:

  ```
  dumpsys --skip {{сервис}}
  ```

- Задать время ожидания в секундах (по умолчанию 10 сек):

  ```
  dumpsys -t {{секунды}}
  ```

# getprop

Показывает информацию о характеристиках системы Android.

Больше информации: https://manned.org/getprop.

- Показать информацию о характеристиках системы Android:

```
getprop
```

- Показать информации о конкретной характеристике:

```
getprop {{prop}}
```

- Показать на уровне SDK API:

```
getprop {{ro.build.version.sdk}}
```

- Показать версию Android:

```
getprop {{ro.build.version.release}}
```

- Показать модель устройства Android:

```
getprop {{ro.vendor.product.model}}
```

- Показать статус блокировки OEM:

```
getprop {{ro.oem_unlock_supported}}
```

- Показать MAC адрес Wi-Fi карты Android:

```
getprop {{ro.boot.wifimacaddr}}
```

# input

Отправить коды событий или жесты сенсорного экрана на устройство Android.

Эту команду можно использовать только через **`adb shell`**.

Больше информации: https://developer.android.com/reference/android/view/KeyEvent.html#constants_1.

- Отправить код события для одного символа на устройство Android:

  ```
  input keyevent {{код_события}}
  ```

- Отправить текст на устройство Android (`%s` означает пробел):

  ```
  input text "{{текст}}"
  ```

- Отправить одно нажатие на экран на устройство Android:

  ```
  input tap {{x_позиция}} {{y_позиция}}
  ```

- Отправить жест смахивания на устройство Android:

  ```
  input swipe {{x_начало}} {{y_начало}} {{x_конец}} {{y_конец}} {{продолжительность_в_мс}}
  ```

- Отправить длинное нажатие на экран на устройство Android с помощью жеста смахивания:

  ```
  input swipe {{x_position}} {{y_position}} {{x_position}} {{y_pos}} {{продолжительность_в_мс}}
  ```

# logcat

Выводит лог системных сообщений, включая трассировки стека при возникновении ошибок и информационные сообщения, записанные приложениями.

Больше информации: https://developer.android.com/tools/logcat.

- Отобразить системные логи:

```
logcat
```

- Записать системные логи в файл:

```
logcat -f {{путь/к/файлу}}
```

- Показать строки, соответствующие регулярному выражению:

```
logcat --regex {{регулярное_выражение}}
```

- Отобразить логи для процесса с указанным идентификатором процесса (PID):

```
logcat --pid {{pid}}
```

- Отобразить логи для процесса указанного пакета:

```
logcat --pid $(pidof -s {{пакет}})
```

# pkg

Утилита управления пакетами для Termux.

Больше информации: https://wiki.termux.com/wiki/Package_Management.

- Обновить все установленные пакеты:

  ```
  pkg upgrade
  ```

- Установить пакет:

  ```
  pkg install {{пакет}}
  ```

- Удалить пакет:

  ```
  pkg uninstall {{пакет}}
  ```

- Переустановить пакет:

  ```
  pkg reinstall {{пакет}}
  ```

- Поиск пакета:

  ```
  pkg search {{пакет}}
  ```

# pm

Показать информацию о приложениях на устройстве Android.

Больше информации: https://developer.android.com/tools/adb#pm.

- Показать список всех установленных приложений:

  ```
  pm list packages
  ```

- Показать список всех установленных системных приложений:

  ```
  pm list packages -s
  ```

- Показать список всех установленных сторонних приложений:

  ```
  pm list packages -3
  ```

- Показать список приложений по ключевым словам:

  ```
  pm list packages {{ключевые_слова}}
  ```

- Показать путь к APK определенного приложения:

  ```
  pm path {{приложение}}
  ```

# screencap

Сделать снимок экрана мобильного дисплея.

Эту команду можно использовать только через **`adb shell`**.

Больше информации: https://developer.android.com/tools/adb#screencap.

- Сделать снимок экрана:

  ```
  screencap {{путь/к/файлу}}
  ```

# settings

Получить информацию об операционной системе Android.

Больше информации: https://web.archive.org/web/20240525010124/https://adbinstaller.com/commands/adb-shell-settings-5b670d5ee7958178a2955536.

- Показать список настроек в `global`:

  ```
  settings list {{global}}
  ```

- Получить значение определенного параметра:

  ```
  settings get {{global}} {{airplane_mode_on}}
  ```

- Задать значение параметра:

  ```
  settings put {{system}} {{screen_brightness}} {{42}}
  ```

- Удалить конкретную настройку:

  ```
  settings delete {{secure}} {{screensaver_enabled}}
  ```

# wm

Показать информацию об экране Android-устройства.

Эту команду можно использовать только через **`adb shell`**.

Больше информации: https://web.archive.org/web/20240420064706/https://adbinstaller.com/commands/adb-shell-wm-5b672b17e7958178a2955538.

- Показать физический размер экрана Android-устройства:

  ```
  wm size
  ```

- Показать физическую плотность экрана Android-устройства:

  ```
  wm density
  ```

# Common

# !

Выполнять подстановку команд из истории оболочки в **sh**, Bash, Zsh, **rbash** и **ksh**.

Больше информации: https://gnu.org/software/bash/manual/bash.html#Event-Designators.

- Подставить предыдущую команду и выполнить её с `sudo`:

```
sudo !!
```

- Подставить команду по номеру её строки, найденному с помощью `history`:

```
!{{номер}}
```

- Подставить команду, использованную указанное количество строк назад:

```
!-{{номер}}
```

- Подставить последнюю команду, которая начинается с указанной строки:

```
!{{строка}}
```

- Подставить аргументы последней команды:

```
{{команда}} !*
```

- Подставить последний аргумент последней команды:

```
{{команда}} !$
```

- Подставить последнюю команду, но без её последнего аргумента:

```
!:-
```

- Вывести последнюю команду, которая начинается со строки, без её выполнения:

```
!{{строка}}:p
```

# 7z

Архиватор файлов с высокой степенью сжатия.

Больше информации: https://manned.org/7z.

- Архивировать ([a]rchive) файл или папку:

```
7z a {{путь/до/архива.7z}} {{путь/до/файла_или_папки}}
```

- Зашифровать существующий архив (включая имена файлов):

```
7z a {{путь/до/зашифрованного_архива.7z}} -p{{пароль}} -
mhe=on {{путь/до/архива.7z}}
```

- Распаковать (e[x]tract) существующий архив, сохраняя оригинальную структуру папок:

```
7z x {{путь/до/архива.7z}}
```

- Распаковать (e[x]tract) архив в нужную папку:

```
7z x {{путь/до/архива.7z}} -o{{путь/до/папки}}
```

- Распаковать (e[x]tract) архив в `stdout`:

```
7z x {{путь/до/архива.7z}} -so
```

- Архивировать ([a]rchive), используя определённый тип архива:

```
7z a -t{{7z|bzip2|gzip|lzip|tar|zip}} {{путь/до/архива}}
{{путь/до/файла_или_папки}}
```

- Вывести ([l]ist) содержимое архива:

```
7z l {{путь/до/архива.7z}}
```

# 7za

Архиватор файлов с высокой степенью сжатия.

То же, что и **7z**, за исключением того, что поддерживает меньшее количество типов файлов, но является кроссплатформенным.

Больше информации: https://manned.org/7za.

- Архивировать ([a]rchive) файл или папку:

```
7za a {{путь/до/архива.7z}} {{путь/до/файла_или_папки}}
```

- Зашифровать существующий архив (включая имена файлов):

```
7za a {{путь/до/зашифрованного_архива.7z}} -p{{пароль}} -
mhe=on {{путь/до/архива.7z}}
```

- Распаковать (e[x]tract) существующий архив, сохраняя оригинальную структуру папок:

```
7za x {{путь/до/архива.7z}}
```

- Распаковать (e[x]tract) архив в нужную папку:

```
7za x {{путь/до/архива.7z}} -o{{путь/до/папки}}
```

- Распаковать (e[x]tract) архив в `stdout`:

```
7za x {{путь/до/архива.7z}} -so
```

- Архивировать ([a]rchive), используя определённый тип архива:

```
7za a -t{{7z|bzip2|gzip|lzip|tar|zip}} {{путь/до/архива.7z}}
{{путь/до/файла_или_папки}}
```

- Вывести ([l]ist) содержимое архива:

```
7za l {{путь/до/архива.7z}}
```

# 7zr

Архиватор файлов с высокой степенью сжатия.

То же, что и **`7z`**, но поддерживает только файлы 7z.

Больше информации: https://manned.org/7zr.

- Архивировать ([a]rchive) файл или папку:

```
7zr a {{путь/до/архива.7z}} {{путь/до/файла_или_папки}}
```

- Зашифровать существующий архив (включая имена файлов):

```
7zr a {{путь/до/зашифрованного_архива.7z}} -p{{пароль}} -
mhe={{on}} {{путь/до/архива.7z}}
```

- Распаковать (e[x]tract) существующий архив, сохраняя оригинальную структуру папок:

```
7zr x {{путь/до/архива.7z}}
```

- Распаковать (e[x]tract) архив в нужную папку:

```
7zr x {{путь/до/архива.7z}} -o{{путь/до/папки}}
```

- Распаковать (e[x]tract) архив в `stdout`:

```
7zr x {{путь/до/архива.7z}} -so
```

- Вывести ([l]ist) содержимое архива:

```
7zr l {{путь/до/архива.7z}}
```

# [

Проверять типы файлов и сравнивать значения.

Возвращает код завершения 0, если условие истинно, и 1, если оно ложно.

Больше информации: https://gnu.org/software/bash/manual/bash.html#index-test.

- Проверить, равна ли указанная переменная заданной строке или нет:

```
[ "${{переменная}}" {{=|!=}} "{{строка}}" ]
```

- Проверить, является ли указанная переменная равной [eq]/не равной [ne]/большей [gt]/меньшей [lt]/большей или равной [ge]/меньшей или равной [le] указанному числу:

```
[ "${{переменная}}" -{{eq|ne|gt|lt|ge|le}} {{целое_число}} ]
```

- Проверить, имеет ли указанная переменная [n]епустое значение:

```
[ -n "${{переменная}}" ]
```

- Проверить, имеет ли указанная переменная пустое значение (нулевой [z] длины):

```
[ -z "${{переменная}}" ]
```

- Проверить, существует ли указанный [f]айл:

```
[ -f {{путь/к/файлу}} ]
```

- Проверить, существует ли указанный каталог ([d]irectory):

```
[ -d {{путь/к/каталогу}} ]
```

- Проверить, существует ли ([e]xists) указанный файл или каталог:

```
[ -e {{путь/к/файлу_или_каталогу}} ]
```

# ]

Завершает условное выражение, начатое с **[**.

- Посмотреть документацию для команды `[`:

```
tldr [
```

# aapt

Утилита для упаковки ресурсов для Android.

Компилирует и упаковывает ресурсы приложений Android.

Больше информации: https://manned.org/aapt.

- Вывести список файлов содержащихся в APK-архиве:

```
aapt list {{путь/до/приложения.apk}}
```

- Отобразить мета-данные приложения (версия, разрешения, и т.д.):

```
aapt dump badging {{путь/до/приложения.apk}}
```

- Создать новый APK-архив с файлами из указанной папки:

```
aapt package -F {{путь/до/приложения.apk}} {{путь/до/папки}}
```

# ab

Утилита бенчмаркинга Apache. Самая простая утилита для проведения нагрузочного тестирования.

Больше информации: https://httpd.apache.org/docs/current/programs/ab.html.

- Запустить 100 запросов HTTP GET по заданному URL:

```
ab -n 100 {{url}}
```

- Запустить 100 запросов HTTP GET, обрабатывая до 10 одновременно, по заданному URL:

```
ab -n 100 -c 10 {{url}}
```

- Запустить 100 запросов HTTP POST по заданному URL, используя в качестве полезной нагрузки JSON из файла:

```
ab -n 100 -T {{application/json}} -p {{путь/до/файла.json}} {{url}}
```

- Использовать постоянное соединение (keep-alive):

```
ab -k {{url}}
```

- Задать максимальное число секунд, которое можно затратить на бенчмаркинг:

```
ab -t {{60}} {{url}}
```

# abduco

Менеджер сессий терминала.

Больше информации: https://manned.org/abduco.

- Вывести список сеансов:

  ```
  abduco
  ```

- Подключиться к сеансу, и создать его, если он не существует:

  ```
  abduco -A {{имя}} {{bash}}
  ```

- Подключиться к сеансу с `dvtm`, и создать его, если он не существует:

  ```
  abduco -A {{имя}}
  ```

- Отключиться от сеанса:

  ```
  <Ctrl \>
  ```

- Подключиться к сеансу в режиме только для чтения:

  ```
  abduco -Ar {{имя}}
  ```

# ack

Утилита для поиска, подобная grep, оптимизирования для программистов.

Смотри также: **`rg`**, которая гораздо быстрее.

Больше информации: https://beyondgrep.com/documentation.

- Найти файлы, содержащие строку или регулярное выражение, рекурсивно в текущей директории:

```
ack "{{шаблон_поиска}}"
```

- Искать по шаблону без учёта регистра:

```
ack {{[-i|--ignore-case]}} "{{шаблон_поиска}}"
```

- Искать строки, соответствующие шаблону, печатая только ([o]nly) совпавший текст, а не остальную часть строки:

```
ack {{[-o|--output '$&']}} "{{шаблон_поиска}}"
```

- Ограничить поиск только файлами определённого типа:

```
ack {{[-t|--type]}} {{ruby}} "{{шаблон_поиска}}"
```

- Не искать в файлах определённого типа:

```
ack {{[-t|--type]}} no{{ruby}} "{{шаблон_поиска}}"
```

- Подсчитать общее количество найденных совпадений:

```
ack {{[-c|--count]}} {{[-h|--no-filename]}}
"{{шаблон_поиска}}"
```

- Вывести только имена файлов и количество совпадений для каждого файла:

```
ack {{[-c|--count]}} {{[-l|--files-with-matches]}}
"{{шаблон_поиска}}"
```

- Вывести все значения, которые можно использовать с `--type`:

```
ack --help-types
```

# act

Запуск GitHub Actions локально с использованием Docker.

Больше информации: https://manned.org/act.

- Вывести список доступных actions:

  ```
  act -l
  ```

- Запустить событие по умолчанию:

  ```
  act
  ```

- Запустить заданное событие:

  ```
  act {{тип_события}}
  ```

- Запустить заданный action:

  ```
  act -a {{action_id}}
  ```

- Не производить реальный запуск actions (пробный прогон):

  ```
  act -n
  ```

- Отображать расширенный лог:

  ```
  act -v
  ```

# adb install

Android Debug Bridge Install: Установка пакетов на эмулятор Android или подключённое устройство Android.

Больше информации: https://developer.android.com/tools/adb.

- Установить приложение Android на эмулятор/устройство:

  ```
  adb install {{путь/до/файла.apk}}
  ```

- Установить приложение Android на конкретный эмулятор/устройство (отменяет использование `$ANDROID_SERIAL`):

  ```
  adb -s {{серийный_номер}} install {{путь/до/файла.apk}}
  ```

- Переустановить существующее приложение, оставив его данные:

  ```
  adb install -r {{путь/до/файла.apk}}
  ```

- Установить приложение Android, разрешив понижение версии (только для отлаживаемых пакетов):

  ```
  adb install -d {{путь/до/файла.apk}}
  ```

- Дать все разрешения, перечисленные в манифесте приложения:

  ```
  adb install -g {{путь/до/файла.apk}}
  ```

- Быстрое обновление установленного пакета путём обновления только тех частей APK, которые изменились:

  ```
  adb install --fastdeploy {{путь/до/файла.apk}}
  ```

# adb reverse

Android Debug Bridge Reverse: обратное соединение от эмулятора Android или подключенного устройства Android.

Больше информации: https://developer.android.com/tools/adb.

- Вывести список всех обратных соединений от эмуляторов и устройств:

  ```
  adb reverse --list
  ```

- Создать обратное соединение по TCP-порту от эмулятора или устройства до localhost:

  ```
  adb reverse tcp:{{удалённый_порт}} tcp:{{локальный_порт}}
  ```

- Удалить обратное соединение из эмулятора или устройства:

  ```
  adb reverse --remove tcp:{{удалённый_порт}}
  ```

- Удалить все обратные соединения на всех эмуляторах и устройствах:

  ```
  adb reverse --remove-all
  ```

# adb shell

Android Debug Bridge Shell: Запуск удалённой командной оболочки на эмуляторе Android или подключенном устройстве Android.

Больше информации: https://developer.android.com/tools/adb.

- Запустить удалённую интерактивную оболочку на эмуляторе или устройстве:

```
adb shell
```

- Получить все свойства от эмулятора или устройства:

```
adb shell getprop
```

- Вернуть всем разрешениям значение по умолчанию:

```
adb shell pm reset-permissions
```

- Отозвать опасные разрешения для приложения:

```
adb shell pm revoke {{пакет}} {{разрешения}}
```

- Вызвать событие клавиши:

```
adb shell input keyevent {{код_клавиши}}
```

- Очистить данные приложения на эмуляторе или устройстве:

```
adb shell pm clear {{пакет}}
```

- Запустить activity на эмуляторе или устройстве:

```
adb shell am start -n {{пакет}}/{{активность}}
```

- Запустить базовый activity на эмуляторе или устройстве:

```
adb shell am start -W -c android.intent.category.HOME -a
android.intent.action.MAIN
```

# adb

Android Debug Bridge: управление запущенным эмулятором Android или подключенным устройством Android.

Некоторые подкоманды, такие как **`shell`**, имеют собственную документацию по использованию.

Больше информации: https://developer.android.com/tools/adb.

- Проверить, запущен ли процесс сервера adb и запустить его:

```
adb start-server
```

- Завершить процесс сервера adb:

```
adb kill-server
```

- Запустить удалённую оболочку на целевом эмуляторе/устройстве:

```
adb shell
```

- Установить приложение Android на эмуляторе/устройстве:

```
adb install -r {{путь/до/файла.apk}}
```

- Скопировать файл/папку с целевого устройства:

```
adb pull {{путь/до/папки_или_файла_на_устройстве}} {{путь/до/локальной_папки}}
```

- Скопировать файл/папку на целевое устройство:

```
adb push {{путь/до/локального_файла_или_папки}} {{путь/до/целевой_папки_на_устройстве}}
```

- Вывести список подключенных устройств:

```
adb devices
```

# AdGuardHome

Программное обеспечение для блокировки рекламы и отслеживания во всей сети.

Больше информации: https://github.com/AdguardTeam/AdGuardHome.

- Запустить AdGuard Home:

  ```
  AdGuardHome
  ```

- Запустить AdGuard с заданной конфигурацией:

  ```
  AdGuardHome --config {{путь/до/AdGuardHome.yaml}}
  ```

- Установить рабочую папку, где будут сохранятья данные:

  ```
  AdGuardHome --work-dir {{путь/до/папки}}
  ```

- Установить или удалить AdGuard Home как службу:

  ```
  AdGuardHome --service {{install|uninstall}}
  ```

- Запустить службу AdGuard Home:

  ```
  AdGuardHome --service start
  ```

- Перезагрузить конфигурацию для службы AdGuard Home:

  ```
  AdGuardHome --service reload
  ```

- Остановить или перезапустить службу AdGuard Home:

  ```
  AdGuardHome --service {{stop|restart}}
  ```

# ag

The Silver Searcher. Аналог **ack**, позиционирующийся как более быстрый.

Больше информации: https://manned.org/ag.

- Найти файлы, содержащие "foo", и вывести подходящие строки с контекстом:

```
ag foo
```

- Найти файлы, содержащие "foo", в заданной папке:

```
ag foo {{путь/к/каталогу}}
```

- Найти файлы, содержащие "foo", но вывести только имена файлов:

```
ag {{[-l|--files-with-matches]}} foo
```

- Найти файлы, содержащие "FOO", независимо от регистра, и вывести только совпадения, а не строки целиком:

```
ag {{[-i|--ignore-case]}} {{[-o|--only-matching]}} FOO
```

- Найти "foo" в файлах, у которых в имени есть "bar":

```
ag foo {{[-G|--file-search-regex]}} bar
```

- Найти файлы, содержимое которых совпадает с регулярным выражением:

```
ag '{{^ba(r|z)$}}'
```

- Найти файлы, у которых имя содержит "foo":

```
ag {{[-g|--filename-pattern]}} foo
```

# alias

Создает псевдонимы -- слова, которые заменяются командой.

Срок действия псевдонима истекает с окончанием текущей сессии командной строки, если только не определить его в конфигурационном файле, например: **`~/.bashrc`** для Bash или **`~/.zshrc`** для Zsh.

Смотрите также: **`unalias`**.

Больше информации: https://www.gnu.org/software/bash/manual/bash.html#index-alias.

- Вывести список всех псевдонимов:

```
alias
```

- Создать типовой псевдоним:

```
alias {{псевдоним}}="{{команда}}"
```

- Посмотреть команду, сопоставленную с данным псевдонимом:

```
alias {{псевдоним}}
```

- Удалить псевдоним:

```
unalias {{псевдоним}}
```

- Сделать `rm` интерактивной командой:

```
alias {{rm}}="{{rm --interactive}}"
```

- Создать `la` как сокращение для `ls --all`:

```
alias {{la}}="{{ls --all}}"
```

# asciidoctor

Преобразователь AsciiDoc файлов в другие форматы для публикации.

Больше информации: https://docs.asciidoctor.org.

- Преобразовать данный `.adoc` файл в HTML (выходной формат по умолчанию):

  ```
  asciidoctor {{путь/к/файлу.adoc}}
  ```

- Преобразовать данный `.adoc` файл в HTML и привязать таблицу стилей CSS:

  ```
  asciidoctor {{[-a|--attribute]}} stylesheet={{путь/к/
  таблице_стилей.css}} {{путь/к/файлу.adoc}}
  ```

- Преобразовать данный `.adoc` файл во встраиваемый HTML, убрав всё, кроме самого текста:

  ```
  asciidoctor {{[-e|--embedded]}} {{путь/к/файлу.adoc}}
  ```

- Преобразовать данный `.adoc` файл в PDF с помощью библиотеки `asciidoctor-pdf`:

  ```
  asciidoctor {{[-b|--backend]}} pdf {{[-r|--require]}}
  asciidoctor-pdf {{путь/к/файлу.adoc}}
  ```

# aspell

Интерактивная проверка орфографии.

Больше информации: http://aspell.net/man-html/index.html.

- Проверить орфографию в одном файле:

```
aspell check {{путь/к/файлу}}
```

- Вывести список неверно написанных слов из стандартного ввода:

```
cat {{путь/к/файлу}} | aspell list
```

- Показать доступные словари:

```
aspell dicts
```

- Запустить `aspell` с использованием другого языка (двухсимвольный код согласно ISO 639):

```
aspell --lang {{cs}}
```

- Вывести список неверно написанных слов из стандартного ввода, игнорируя слова из персонального списка:

```
cat {{путь/к/файлу}} | aspell --personal
{{персональный_список_слов.pws}} list
```

# bash

Bourne-Again SHell, **sh**-совместимый командный интерпретатор.

Смотрите также: **zsh**, **histexpand** (подстановка из истории).

Больше информации: https://www.gnu.org/software/bash/manual/bash.html#Invoking-Bash.

- Запустить интерактивную сессию оболочки:

```
bash
```

- Запустить интерактивную сессию оболочки без загрузки файлов конфигурации:

```
bash --norc
```

- Выполнить указанные команды [c]:

```
bash -c "{{echo 'bash выполняется'}}"
```

- Выполнить указанный скрипт:

```
bash {{путь/к/скрипту.sh}}
```

- Выполнить [x] указанный скрипт, выводя каждую команду перед её выполнением:

```
bash -x {{путь/к/скрипту.sh}}
```

- Выполнить указанный скрипт и остановиться при первой ошибке [e]:

```
bash -e {{путь/к/скрипту.sh}}
```

- Выполнить указанные команды, полученные из `stdin`:

```
{{echo "echo 'bash выполняется'"}} | bash
```

- Запустить ограниченную [r] сессию оболочки:

```
bash -r
```

# bg

Возобновляет работу приостановленного задания (например, с помощью **`<Ctrl z>`**) и оставляет его работать в фоне.

Смотрите также: **`jobs`**, **`fg`**, **`disown`**.

Больше информации: https://www.gnu.org/software/bash/manual/bash.html#index-bg.

- Возобновить работу последнего приостановленного задания и продолжить его выполнение в фоне:

  ```
  bg
  ```

- Возобновить указанное задание и продолжить его выполнение в фоне (используйте `jobs`, чтобы получить его идентификатор):

  ```
  bg %{{идентификатор_задания}}
  ```

# cabal

Интерфейс командной строки для инфраструктуры пакетов Haskell (Cabal).

Управление Haskell-проектами и Cabal-пакетами из репозитория Hackage.

Больше информации: https://cabal.readthedocs.io/en/latest/getting-started.html.

- Искать и вывести список пакетов из Hackage:

  ```
  cabal list {{строка_поиска}}
  ```

- Показать информацию о пакете:

  ```
  cabal info {{имя_пакета}}
  ```

- Скачать и установить пакет:

  ```
  cabal install {{имя_пакета}}
  ```

- Создать новый Haskell-проект в текущей папке:

  ```
  cabal init
  ```

- Собрать проект в текущей папке:

  ```
  cabal build
  ```

- Запустить тесты из проекта в текущей папке:

  ```
  cabal test
  ```

# cat

Выводит и объединяет файлы.

Больше информации: https://manned.org/cat.1posix.

- Выводит содержимое файла в стандартный вывод:

```
cat {{путь/к/файлу}}
```

- Объединяет несколько файлов в один итоговый файл:

```
cat {{путь/к/файлу1 путь/к/файлу2 ...}} > {{путь/к/
итоговому_файлу}}
```

- Добавляет несколько файлов в конец итогового файла:

```
cat {{путь/к/файлу1 путь/к/файлу2 ...}} >> {{путь/к/
итоговому_файлу}}
```

- Копирует содержимое файла в итоговый файл без буферизации:

```
cat -u {{/dev/tty12}} > {{/dev/tty13}}
```

- Записывает данные из стандартного ввода в файл:

```
cat - > {{путь/к/файлу}}
```

# cd

Изменить текущий рабочий каталог.

Больше информации: https://www.gnu.org/software/bash/manual/bash.html#index-cd.

- Перейти в указанный каталог:

```
cd {{путь/к/каталогу}}
```

- Перейти в каталог выше:

```
cd ..
```

- Перейти в домашний каталог текущего пользователя:

```
cd
```

- Перейти в домашний каталог указанного пользователя:

```
cd ~{{имя_пользователя}}
```

- Перейти в ранее выбранный каталог:

```
cd -
```

- Перейти в корневой каталог:

```
cd /
```

# chmod

Изменить права доступа файлу или папке.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/chmod-invocation.html.

- Дать пользователю ([u]ser), который владеет файлом, права на его исполнение (e[x]ecute):

```
chmod u+x {{файл}}
```

- Дать права пользователю ([u]ser) права чтения ([r]ead) и записи ([w]rite) в файл/папку:

```
chmod u+rw {{файл_или_папка}}
```

- Убрать права на исполнение (e[x]ecute) у группы ([g]roup):

```
chmod g-x {{файл}}
```

- Дать всем ([a]ll) пользователям права на чтение ([r]ead) и исполнение (e[x]ecute):

```
chmod a+rx {{файл}}
```

- Дать другим ([o]thers) (не из группы владельцев файла) такие же права, как и у группы ([g]roup):

```
chmod o=g {{файл}}
```

- Убрать все права у других ([o]thers):

```
chmod o= {{файл}}
```

- Изменить права рекурсивно, дав группе ([g]roup) и другим ([o]thers) возможность записи ([w]rite) в папку:

```
chmod {{[-R|--recursive]}} g+w,o+w {{папка}}
```

- Рекурсивно дать для всех ([a]ll) пользователей права на чтение ([r]ead) файлов и права на исполнение (e[X]ecute) поддиректорий внутри указанной директории:

```
chmod {{[-R|--recursive]}} a+rX {{папка}}
```

# clang-cpp

Эта команда — псевдоним для **`clang++`**.

- Смотри документацию для оригинальной команды:

```
tldr clang++
```

# clojure

Эта команда — псевдоним для **`clj`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr clj
  ```

# cola

Эта команда — псевдоним для **`git-cola`**.

- Смотри документацию для оригинальной команды:

```
tldr git-cola
```

# cut

Вырезать поля из стандартного ввода или файлов.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/cut-invocation.html.

- Вывести указанный диапазон символов/полей каждой строки (`--characters|fields 1|1,10|1-10|1-|-10` далее обозначается как `диапазон`):

```
{{команда}} | cut --{{characters|fields}} {{1|1,10|1-10|1-|-10}}
```

- Вывести диапазон полей каждой строки с указанным разделителем:

```
{{команда}} | cut --delimiter "{{,}}" --fields {{1}}
```

- Вывести диапазон символов каждой строки указанного файла:

```
cut --characters {{1}} {{путь/к/файлу}}
```

# cwebp

Сжимает файл изображения в формат WebP.

Больше информации: https://developers.google.com/speed/webp/docs/cwebp.

- Сжать WebP со стандартными настройками (q = 75) с сохранением в выходной файл:

```
cwebp {{путь/к/изображению}} -o {{путь/к/результату.webp}}
```

- Сжать WebP с наилучшим качеством и наибольшим размером файла:

```
cwebp {{путь/к/изображению}} -o {{путь/к/результату.webp}} -q
{{100}}
```

- Сжать WebP с наихудшим качеством и наименьшим размером файла:

```
cwebp {{путь/к/изображению}} -o {{путь/к/результату.webp}} -q
{{0}}
```

- Сжать WebP с изменением размера изображения:

```
cwebp {{путь/к/изображению}} -o {{путь/к/результату.webp}} -
resize {{width}} {{height}}
```

- Сжать WebP с удалением информации о прозрачности:

```
cwebp {{путь/к/изображению}} -o {{путь/к/результату.webp}} -
noalpha
```

# df

Отображать информацию об использовании дискового пространства файловых систем.

Больше информации: https://manned.org/df.1posix.

- Показывать все файловые системы и использование диска для них в 512-байтных блоках:

```
df
```

- Показывать информацию о файловой системе для указанного файла или каталога:

```
df {{путь/к/файлу_или_каталогу}}
```

- Использовать 1024-байтные блоки при отображении размеров:

```
df -k
```

- Показывать информацию в портируемом (POSIX) формате:

```
df -P
```

# dig

Утилита для поиска информации в DNS.

Больше информации: https://manned.org/dig.

- Искать IP-адрес(а), связанные с именем хоста (A-записи):

```
dig +short {{example.com}}
```

- Получить подробный ответ для указанного домена (A-записи):

```
dig +noall +answer {{example.com}}
```

- Запросить определенный тип DNS-записи, связанный с указанным доменным именем:

```
dig +short {{example.com}} {{A|MX|TXT|CNAME|NS}}
```

- Указать альтернативный DNS-сервер для запроса и, опционально, использовать DNS поверх TLS (DoT):

```
dig {{+tls}} @{{1.1.1.1|8.8.8.8|9.9.9.9|...}} {{example.com}}
```

- Выполнить обратный DNS-запрос для IP-адреса (PTR-запись):

```
dig -x {{8.8.8.8}}
```

- Найти авторитативные серверы имен для зоны и отобразить SOA-записи:

```
dig +nssearch {{example.com}}
```

- Выполнить итеративные запросы и отобразить полный путь трассировки для разрешения доменного имени:

```
dig +trace {{example.com}}
```

- Запросить DNS-сервер через нестандартный порт [p], используя протокол TCP:

```
dig +tcp -p {{порт}} @{{dns_сервер}} {{example.com}}
```

# docker

Управление контейнерами и образами Docker.

Некоторые подкоманды, такие как **`run`**, имеют собственную документацию по использованию.

Больше информации: https://docs.docker.com/reference/cli/docker/.

- Список всех контейнеров Docker (запущенных и остановленных):

  ```
  docker ps {{[-a|--all]}}
  ```

- Запустить контейнера из образа с заданным именем:

  ```
  docker run --name {{имя_контейнера}} {{имя_образа}}
  ```

- Запустить или остановить существующий контейнер:

  ```
  docker {{start|stop}} {{имя_контейнера}}
  ```

- Загрузить образ из репозитория Docker:

  ```
  docker pull {{имя_образа}}
  ```

- Показать список уже загруженных образов:

  ```
  docker images
  ```

- Запустить интерактивный ([i]) телетайп ([t]) с командной оболочкой Bourne shell (sh) внутри запущенного контейнера:

  ```
  docker exec {{[-it|--interactive --tty]}} {{имя_контейнера}} {{sh}}
  ```

- Удалить остановленный контейнер:

  ```
  docker rm {{имя_контейнера}}
  ```

- Получить и следить за логами контейнера:

  ```
  docker logs {{[-f|--follow]}} {{имя_контейнера}}
  ```

# DOOM

Классический шутер с открытым исходным кодом, ориентированный на моддинг и многопользовательский режим.

Вы можете заменить **`zandronum-server`** на практически любой порт DOOM.

Больше информации: https://doomwiki.org/wiki/Source_port_parameters.

- Запустить кооперативный многопользовательский режим:

```
{{zandronum-server}} -iwad {{wad}} +map {{карта}} -host
{{число_игроков}}
```

- Запустить многопользовательский режим "deathmatch":

```
{{zandronum-server}} -iwad {{wad}} +map {{карта}} -host
{{число_игроков}} -deathmatch
```

# dotnet build

Собирает приложение .NET и все его зависимости.

Больше информации: https://learn.microsoft.com/dotnet/core/tools/dotnet-build.

- Скомпилировать проект или решение в текущем каталоге:

```
dotnet build
```

- Скомпилировать проект или решение .NET в режиме debug:

```
dotnet build {{путь/к/проекту_или_решению}}
```

- Скомпилировать в режиме release:

```
dotnet build {{[-c|--configuration]}} {{Release}}
```

- Скомпилировать без восстановления зависимостей:

```
dotnet build --no-restore
```

- Скомпилировать с заданным уровнем детализации выводимой информации:

```
dotnet build {{[-v|--verbosity]}} {{quiet|minimal|normal|
detailed|diagnostic}}
```

- Скомпилировать для заданной среды исполнения:

```
dotnet build {{[-r|--runtime]}}
{{идентификатор_среды_исполнения}}
```

- Указать целевой каталог:

```
dotnet build {{[-o|--output]}} {{путь/к/каталогу}}
```

# dotnet publish

Публикует .NET-приложение и его зависимости в каталог для развёртывания на целевой системе.

Больше информации: https://learn.microsoft.com/dotnet/core/tools/dotnet-publish.

- Скомпилировать проект .NET в режиме release:

```
dotnet publish {{[-c|--configuration]}} Release {{путь/к/файлу_проекта}}
```

- Опубликовать ваше приложение с заданной средой исполнения .NET Core:

```
dotnet publish {{[-sc|--self-contained]}} true {{[-r|--runtime]}} {{идентификатор_среды_исполнения}} {{путь/к/файлу_проекта}}
```

- Упаковать приложение в один исполняемый файл для заданной платформы:

```
dotnet publish {{[-r|--runtime]}} {{идентификатор_среды_исполнения}} -p:PublishSingleFile=true {{путь/к/файлу_проекта}}
```

- Обрезать неиспользуемые библиотеки чтобы уменьшить размер развёртывания приложения:

```
dotnet publish {{[-sc|--self-contained]}} true {{[-r|--runtime]}} {{идентификатор_среды_исполнения}} -p:PublishTrimmed=true {{путь/к/файлу_проекта}}
```

- Скомпилировать проект .NET без восстановления зависимостей:

```
dotnet publish --no-restore {{путь/к/файлу_проекта}}
```

- Указать целевой каталог:

```
dotnet publish {{[-o|--output]}} {{путь/к/каталогу}} {{путь/к/файлу_проекта}}
```

# dotnet restore

Восстанавливает зависимости и утилиты для проекта .NET.

Больше информации: https://learn.microsoft.com/dotnet/core/tools/dotnet-restore.

- Восстановить зависимости для проекта или решения .NET в текущем каталоге:

  ```
  dotnet restore
  ```

- Восстановить зависимости для проекта или решения .NET по заданному пути:

  ```
  dotnet restore {{путь/к/проекту_или_решению}}
  ```

- Восстановить зависимости без кеширования HTTP-запросов:

  ```
  dotnet restore --no-cache
  ```

- Принудительно разрешать все зависимости, даже если предыдущее восстановление было успешным:

  ```
  dotnet restore --force
  ```

- Восстановить зависимости, рассматривая ошибки источника пакетов как предупреждения:

  ```
  dotnet restore --ignore-failed-sources
  ```

- Восстановить зависимости, используя заданный уровень детализации выводимой информации:

  ```
  dotnet restore {{[-v|--verbosity]}} {{quiet|minimal|normal|detailed|diagnostic}}
  ```

# dotnet

Кросс-платформенная утилита командной строки .NET для .NET Core.

Некоторые подкоманды, такие как **`build`**, имеют собственную документацию по использованию.

Больше информации: https://learn.microsoft.com/dotnet/core/tools.

- Инициализировать новый проект .NET:

  ```
  dotnet new {{короткое_имя_шаблона}}
  ```

- Восстановить пакеты nuget:

  ```
  dotnet restore
  ```

- Собрать и запустить проект .NET в текущей папке:

  ```
  dotnet run
  ```

- Запустить собранное приложение .NET (требуется только среда исполнения, для остальных команд требуется установленный .NET Core SDK):

  ```
  dotnet {{путь/до/приложения.dll}}
  ```

# echo

Выводит указанные аргументы.

Смотрите также: **`printf`**.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/echo-invocation.html.

- Вывести текстовое сообщение. Примечание: кавычки необязательны:

```
echo "{{Привет, мир}}"
```

- Вывести сообщение с переменными окружения:

```
echo "{{Мой путь: $PATH}}"
```

- Вывести сообщение без завершающего символа новой строки:

```
echo -n "{{Привет, мир}}"
```

- Добавить сообщение в файл:

```
echo "{{Привет, мир}}" >> {{файл.txt}}
```

- Включить интерпретацию escape-последовательностей (специальных символов):

```
echo -e "{{Столбец 1\tСтолбец 2}}"
```

- Вывести статус выхода последней выполненной команды (Примечание: в командной строке Windows и PowerShell эквивалентные команды — `echo %errorlevel%` и `$lastexitcode` соответственно):

```
echo $?
```

# ed

Оригинальный текстовый редактор Unix.

Смотрите также: **awk**, **sed**.

Больше информации: https://www.gnu.org/software/ed/manual/ed_manual.html.

- Запустить интерактивную сессию редактора с пустым документом:

```
ed
```

- Запустить интерактивную сессию редактора с пустым документом и указанной подсказкой:

```
ed {{[-p|--prompt]}} '{{> }}'
```

- Запустить интерактивную сессию редактора с удобными для пользователя ошибками:

```
ed {{[-v|--verbose]}}
```

- Запустить интерактивную сессию редактора пустым документом и без диагностики, подсчета байтов и '!' подсказки:

```
ed {{[-q|--quiet]}}
```

- Запустить интерактивную сессию редактора без изменения статуса выхода при сбое команды:

```
ed {{[-l|--loose-exit-status]}}
```

- Редактировать указанный файл (это показывает количество байт загруженного файла):

```
ed {{путь/к/файлу}}
```

- Заменить строку указанной на всех строках:

```
,s/{{регулярное_выражение}}/{{замена}}/g<Enter>
```

# exit

Выйти из оболочки.

Больше информации: https://manned.org/exit.1posix.

- Выход из оболочки с кодом выхода последней выполненной команды:

```
exit
```

- Выйти из оболочки с указанным кодом выхода:

```
exit {{код_выхода}}
```

# fg

Переключение задания на передний план.

Смотрите также: **`jobs`**, **`bg`**, **`disown`**.

Больше информации: https://www.gnu.org/software/bash/manual/bash.html#index-fg.

- Переключить последнее приостановленное или выполняющееся в фоне задание на передний план:

  ```
  fg
  ```

- Переключить указанное задание на передний план:

  ```
  fg %{{идентификатор_задания}}
  ```

# fossil ci

Эта команда — псевдоним для **`fossil commit`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr fossil commit
  ```

# fossil forget

Эта команда — псевдоним для **`fossil rm`**.

Больше информации: https://fossil-scm.org/home/help/forget.

- Смотри документацию для оригинальной команды:

  ```
  tldr fossil rm
  ```

# fossil new

Эта команда — псевдоним для **`fossil init`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr fossil init
  ```

# fossil rm

Эта команда — псевдоним для **`fossil delete`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr fossil delete
  ```

# gh cs

Эта команда — псевдоним для **`gh codespace`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr gh codespace
  ```

# ghc

Компилятор Glasgow Haskell Compiler.

Компиляция и компоновка исходных файлов Haskell.

Больше информации: https://downloads.haskell.org/ghc/latest/docs/users_guide/usage.html.

- Найти и скомпилировать все модули в текущей папке:

```
ghc Main
```

- Скомпилировать один файл:

```
ghc {{файл.hs}}
```

- Скомпилировать с использованием дополнительной оптимизации:

```
ghc -O {{файл.hs}}
```

- Остановить компиляцию после создания объектных файлов (.o):

```
ghc -c {{файл.hs}}
```

- Запустить REPL (интерактивную оболочку):

```
ghci
```

- Вычислить одно выражение:

```
ghc -e {{выражение}}
```

# ghci

Интерактивная среда Glasgow Haskell Compiler.

Больше информации: https://downloads.haskell.org/ghc/latest/docs/html/users_guide/ghci.html.

- Запустить REPL (интерактивную оболочку):

  ```
  ghci
  ```

- Запустить REPL и загрузить указанный исходный файл Haskell:

  ```
  ghci {{исходный_файл.hs}}
  ```

- Запустить REPL и включить опцию языка:

  ```
  ghci -X{{опция_языка}}
  ```

- Запустить REPL и включить определённый уровень предупреждений компилятора (например, `all` или `compact`):

  ```
  ghci -W{{уровень_предупреждений}}
  ```

- Запустить REPL со списком папок, разделённых двоеточием, в которых нужно искать исходные файлы:

  ```
  ghci -i{{путь/до/папки1:путь/до/папки2:...}}
  ```

# ghcup

Установщик набора инструментов Haskell.

Установка, управление и обновление наборов инструментов Haskell.

Больше информации: https://gitlab.haskell.org/haskell/ghcup-hs.

- Запустить интерактивный текстовый интерфейс:

  ```
  ghcup tui
  ```

- Вывести список доступных версий GHC/cabal:

  ```
  ghcup list
  ```

- Установить рекомендуемую версию GHC:

  ```
  ghcup install ghc
  ```

- Установить указанную версию GHC:

  ```
  ghcup install ghc {{версия}}
  ```

- Задать "активную" версию GHC:

  ```
  ghcup set ghc {{версия}}
  ```

- Установить инструмент cabal-install:

  ```
  ghcup install cabal
  ```

- Обновить сам `ghcup`:

  ```
  ghcup upgrade
  ```

# gimp

GNU программа для работы с изображениями.

Смотрите также: **krita**.

Больше информации: https://docs.gimp.org/en/gimp-fire-up.html#gimp-concepts-running-command-line.

- Запустить GIMP:

```
gimp
```

- Запустить без заставки:

```
gimp --no-splash
```

- Открыть указанные файлы:

```
gimp --new-instance {{путь/к/изображению1 путь/к/
изображению2 ...}}
```

- Вывести ошибки и предупреждения в консоль, вместо отображения их в диалоговом окне:

```
gimp --console-messages
```

- Включить обработчики сигналов отладки:

```
gimp --debug-handlers
```

# gnmic sub

Эта команда — псевдоним для **`gnmic subscribe`**.

- Смотри документацию для оригинальной команды:

```
tldr gnmic subscribe
```

# go build

Скомпилировать исходники Go.

Больше информации: https://pkg.go.dev/cmd/go#hdr-Compile_packages_and_dependencies.

- Скомпилировать файл с пакетом `main` (выходным файлом будет имя файла без расширения):

```
go build {{путь/к/main.go}}
```

- Скомпилировать с указанием имени выходного файла:

```
go build -o {{путь/к/исполняемому_файлу}} {{путь/к/исходному_файлу.go}}
```

- Скомпилировать пакет:

```
go build -o {{путь/к/исполняемому_файлу}} {{путь/к/пакету}}
```

- Скомпилировать пакет `main` в исполняемый файл, включив обнаружение гонки данных:

```
go build -race -o {{путь/к/исполняемому_файлу}} {{путь/к/пакету_main}}
```

# go test

Тестируйте пакеты Go (файлы должны иметь окончание **`_test.go`**).

Больше информации: https://pkg.go.dev/cmd/go#hdr-Testing_flags.

- Протестировать пакет, находящийся в текущем каталоге:

  ```
  go test
  ```

- Протестировать пакет в текущем каталоге с подробным выводом ([v]erbose):

  ```
  go test -v
  ```

- Протестировать пакеты в текущем каталоге и всех подкаталогах (обратите внимание на `...`):

  ```
  go test -v ./...
  ```

- Протестировать пакет в текущем каталоге и запустить все бенчмарки:

  ```
  go test -v -bench .
  ```

- Протестировать пакет в текущем каталоге и запустить все бенчмарки в течение 50 секунд:

  ```
  go test -v -bench . -benchtime 50s
  ```

- Протестировать пакет с анализом покрытия:

  ```
  go test -cover
  ```

# go

Управляйте исходным кодом Go.

Некоторые подкоманды, такие как **`build`**, имеют собственную документацию по использованию.

Больше информации: https://pkg.go.dev/cmd/go.

- Скачать и установить пакет, указанный по его пути импорта:

  ```
  go get {{путь/к/пакету}}
  ```

- Скомпилировать и запустить исходный файл (он должен содержать пакет `main`):

  ```
  go run {{файл}}.go
  ```

- Скомпилировать исходный файл в исполняемый файл с указанным именем:

  ```
  go build -o {{исполняемый_файл}} {{файл}}.go
  ```

- Скомпилировать пакет, находящийся в текущем каталоге:

  ```
  go build
  ```

- Выполнить все тесты для текущего пакета (файлы должны иметь окончание `_test.go`):

  ```
  go test
  ```

- Скомпилировать и установить текущий пакет:

  ```
  go install
  ```

- Инициализировать новый модуль в текущем каталоге:

  ```
  go mod init {{имя_модуля}}
  ```

# grep

Поиск по шаблону в файлах используя регулярные выражения.

Больше информации: https://www.gnu.org/software/grep/manual/grep.html.

- Искать в файле по шаблону:

```
grep "{{шаблон_поиска}}" {{путь/к/файлу}}
```

- Искать по заданной подстроке (регулярные выражения отключены):

```
grep {{[-F|--fixed-strings]}}
"{{заданная_подстрока}}" {{путь/к/файлу}}
```

- Искать по шаблону во всех файлах в директории рекурсивно, показывая номера строк, там где подстрока была найдена, исключая бинарные(двоичные) файлы:

```
grep {{[-rnI|--recursive --line-number --binary-
files=without-match]}} "{{шаблон_поиска}}" {{путь/к/
директории}}
```

- Искать, используя расширенные регулярные выражения (поддержка `?`, `+`, `{}`, `()`, и `|`), без учета регистра:

```
grep {{[-Ei|--extended-regexp --ignore-case]}}
"{{шаблон_поиска}}" {{путь/к/файлу}}
```

- Вывести 3 строки содержимого, до или после каждого совпадения:

```
grep {{--context|--before-context|--after-context}} 3
"{{шаблон_поиска}}" {{путь/к/файлу}}
```

- Вывести имя файла и номер строки для каждого совпадения:

```
grep {{[-Hn|--with-filename --line-number]}} --color=always
"{{шаблон_поиска}}" {{путь/к/файлу}}
```

- Искать строки, где есть совпадение по шаблону поиска, вывод только совпадающей части текста:

```
grep {{[-o|--only-matching]}} "{{шаблон_поиска}}" {{путь/к/
файлу}}
```

- Искать строки в стандартном потоке ввода которые не совпадают с шаблоном поиска:

```
cat {{путь/к/файлу}} | grep {{[-v|--invert-match]}}
"{{шаблон_поиска}}"
```

# gzip

Сжимать и распаковывать файлы с помощью сжатия **`gzip`** (LZ77).

Больше информации: https://www.gnu.org/software/gzip/manual/gzip.html.

- Сжать файл, заменив его архивом `gzip`:

```
gzip {{путь/к/файлу}}
```

- Распаковать файл, заменив его оригинальной несжатой версией:

```
gzip {{[-d|--decompress]}} {{путь/к/файлу.gz}}
```

- Сжать файл, сохранив исходный файл:

```
gzip {{[-k|--keep]}} {{путь/к/файлу}}
```

- Сжать файл, указав имя выходного файла:

```
gzip {{[-c|--stdout]}} {{путь/к/файлу}} > {{путь/к/
сжатому_файлу.gz}}
```

- Распаковать `gzip` архив, указав имя выходного файла:

```
gzip {{[-cd|--stdout --decompress]}} {{путь/к/файлу.gz}} >
{{путь/к/распакованному_файлу}}
```

- Указать уровень сжатия. 1 - самый быстрый (низкое сжатие), 9 - самый медленный (высокое сжатие), 6 - по умолчанию:

```
gzip -{{1..9}} {{[-c|--stdout]}} {{путь/к/файлу}} > {{путь/к/
сжатому_файлу.gz}}
```

- Показывать имя и процент сжатия для каждого сжатого или распакованного файла:

```
gzip {{[-vd|--verbose --decompress]}} {{путь/к/файлу.gz}}
```

# head

Выводит первую часть файлов.

Больше информации: https://manned.org/head.1p.

- Вывести первые несколько строк из файла:

```
head -n {{количетсво_строк}} {{имя_файла}}
```

- Вывести первые несколько байтов из файла:

```
head -c {{количество_байт}} {{имя_файла}}
```

- Вывести все содержимое файла кроме нескольких последних строк:

```
head -n -{{количетсво_строк}} {{имя_файла}}
```

- Вывести все содержимое файла кроме нескольких последних байт:

```
head -c -{{количество_байт}} {{имя_файла}}
```

# history

История командной строки.

Больше информации: https://www.gnu.org/software/bash/manual/bash.html#index-history.

- Отобразить список истории команд с номерами строк:

  ```
  history
  ```

- Отобразить последние 20 команд (в Zsh отображает все команды, начиная с 20-й):

  ```
  history 20
  ```

- Отобразить историю команд с отметками времени в разных форматах (доступно только в Zsh):

  ```
  history -{{d|f|i|E}}
  ```

- Очистить ([c]lear) список истории команд текущей сессии Bash:

  ```
  history -c
  ```

- Перезаписать (over[w]rite) файл истории текущей историей сессии Bash (часто комбинируется с `history -c` для полной очистки истории):

  ```
  history -w
  ```

- Удалить ([d]elete) элемент истории с указанным номером:

  ```
  history -d {{номер}}
  ```

- Добавить команду в историю, не запуская её:

  ```
  history -s {{команда}}
  ```

- Выполнить команду без добавления её в историю с помощью добавления пробела в начале:

  ```
  <Space>{{команда}}
  ```

# hostname

Показ и изменение системного имени хоста.

Больше информации: https://manned.org/hostname.

- Показать имя хоста:

```
hostname
```

- Показать сетевой адрес, соответствующий имени хоста:

```
hostname -i
```

- Показать все сетевые адреса хоста:

```
hostname -I
```

- Показать полное доменное имя (FQDN, Fully Qualified Domain Name):

```
hostname --fqdn
```

- Задать имя хоста:

```
hostname {{новое_имя}}
```

# hunspell

Проверка орфографии.

Больше информации: https://github.com/hunspell/hunspell.

- Проверить орфографию в указанном файле:

  ```
  hunspell {{путь/до/файла}}
  ```

- Проверить орфографию в указанном файле, используя американский словарь (en_US):

  ```
  hunspell -d {{en_US}} {{путь/до/файла}}
  ```

- Вывести список неправильно написанных слов в файле:

  ```
  hunspell -l {{путь/до/файла}}
  ```

# ispell

Интерактивная проверка орфографии.

Больше информации: https://www.cs.hmc.edu/~geoff/ispell-man.html.

- Начать интерактивную сессию:

  ```
  ispell
  ```

- Проверить на ошибки указанный файл и интерактивно применить исправления:

  ```
  ispell {{путь/до/файла}}
  ```

- Отобразить версию:

  ```
  ispell -v
  ```

# jobs

Отображение статуса заданий в текущей сессии.

Больше информации: https://manned.org/jobs.

- Показать статусы всех заданий:

```
jobs
```

- Показать статус конкретного задания:

```
jobs %{{идентификатор_задания}}
```

- Показать статусы и идентификаторы процесса всех заданий:

```
jobs -l
```

- Показать идентификаторы процесса всех заданий:

```
jobs -p
```

# jq

Процессор JSON, использующий предметно-ориентированный язык (DSL).

Больше информации: https://jqlang.github.io/jq/manual/.

- Выполнить указанное выражение с выводом в цветном и отформатированном виде:

```
jq '.' {{путь/к/файлу.json}}
```

- Выполнить указанный скрипт:

```
{{cat путь/к/файлу.json}} | jq {{[-f|--from-file]}} {{путь/к/
скрипту.jq}}
```

- Передать указанные аргументы:

```
{{cat путь/к/файлу.json}} | jq {{--arg "имя1" "значение1" --
arg "имя2" "значение2" ...}} '{{. + $ARGS.named}}'
```

- Создать новый объект JSON на основе объектов из нескольких файлов:

```
{{cat путь/к/нескольким_json_файлам_*.json}} | jq
'{{{новый_ключ1: .ключ1,
новый_ключ2: .ключ2.вложенный_ключ, ...}}}'
```

- Вывести указанные элементы массива:

```
{{cat путь/к/файлу.json}} | jq '{{.[индекс1], .
[индекс2], ...}}'
```

- Вывести все элементы массива/значения объекта:

```
{{cat путь/к/файлу.json}} | jq '.[]'
```

- Вывести объекты из массива, отфильтровав их по двум условиям:

```
{{cat путь/к/файлу.json}} | jq '.[] |
select((.ключ1=="значение1") and .ключ2=="значение2")'
```

- Добавить/удалить указанные ключи:

```
{{cat путь/к/файлу.json}} | jq '. {{+|-}} {{{"ключ1":
"значение1", "ключ2": "значение2", ...}}}'
```

# krita

Krita - программа для создания эскизов и рисования, разработанная для цифровых художников.

Смотрите также: **`gimp`**.

Больше информации: https://docs.krita.org/en/reference_manual/linux_command_line.html.

- Запустить krita:

```
krita
```

- Открыть указанные файлы:

```
krita {{путь/к/изображению1 путь/к/изображению2 ...}}
```

- Запустить без заставки:

```
krita --nosplash
```

- Запустить с указанным рабочим пространством (`Animation`):

```
krita --workspace {{Animation}}
```

- Запустить в полноэкранном режиме:

```
krita --fullscreen
```

# llvm-ar

Эта команда — псевдоним для **`ar`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr ar
  ```

# llvm-g++

Эта команда — псевдоним для **`clang++`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr clang++
  ```

# llvm-gcc

Эта команда — псевдоним для **`clang`**.

- Смотри документацию для оригинальной команды:

```
tldr clang
```

# llvm-nm

Эта команда — псевдоним для **`nm`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr nm
  ```

# llvm-objdump

Эта команда — псевдоним для **`objdump`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr objdump
  ```

# llvm-strings

Эта команда — псевдоним для **`strings`**.

- Смотри документацию для оригинальной команды:

```
tldr strings
```

# ls

Вывод содержимого каталога.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/ls-invocation.html.

- Список файлов по одному в строке:

```
ls -1
```

- Список всех файлов, включая скрытые:

```
ls {{[-a|--all]}}
```

- Список всех файлов с добавлением в конце `/` к именам директорий:

```
ls {{[-F|--classify]}}
```

- Подробный список с выводом разрешений, владельцев, размера и даты изменения всех файлов:

```
ls {{[-la|--all -l]}}
```

- Подробный список с выводом размера файла в удобочитаемых единицах (КиБ, МиБ, ГиБ):

```
ls {{[-lh|-l --human-readable]}}
```

- Подробный список, отсортированный по размеру файлов (по убыванию):

```
ls {{-lSR|-lS --recursive}}
```

- Подробный список, отсортированный по дате изменения файла (сначала более старые):

```
ls {{[-ltr|-lt --reverse]}}
```

- Список только директорий:

```
ls {{[-d|--directory]}} */
```

# man

Форматирует и отображает страницы руководства.

Больше информации: https://manned.org/man.

- Показать страницу руководства для команды:

```
man {{команда}}
```

- Открыть страницу руководства для команды в браузере (переменная окружения `BROWSER` может заменить `=browser_name`):

```
man {{[-H|--html=]}}{{имя_браузера}} {{команда}}
```

- Показать страницу руководства для команды из раздела 7:

```
man 7 {{команда}}
```

- Показать все доступные разделы для команды:

```
man {{[-f|--whatis]}} {{команда}}
```

- Показать пути, по которым выполняется поиск страниц руководства:

```
man {{[-w|--path]}}
```

- Показать расположение файла страницы руководства, а не саму страницу:

```
man {{[-w|--where]}} {{команда}}
```

- Показать страницу руководства, используя определенную локаль:

```
man {{[-L|--locale]}} {{локаль}} {{команда}}
```

- Искать страницы руководства, содержащие строку поиска:

```
man {{[-k|--apropos]}} "{{строка_поиска}}"
```

# mcs

Mono компилятор C#.

Больше информации: https://manned.org/mcs.1.

- Скомпилировать указанные файлы:

```
mcs {{путь/к/входному_файлу1.cs путь/к/
входному_файлу2.cs ...}}
```

- Указать имя выходной программы:

```
mcs -out:{{путь/к/файлу.exe}} {{путь/к/входному_файлу1.cs
путь/к/входному_файлу2.cs ...}}
```

- Указать тип выходной программы:

```
mcs -target:{{exe|winexe|library|module}} {{путь/к/
входному_файлу1.cs путь/к/входному_файлу2.cs ...}}
```

# micro

Micro — это современный и интуитивно понятный консольный текстовый редактор.

Micro поддерживает клавиатуру и мышь для навигации и/или выделения текста.

Больше информации: https://micro-editor.github.io.

- Открыть файл:

  ```
  micro {{файл}}
  ```

- Сохранить файл:

  ```
  <Ctrl s>
  ```

- Вырезать всю строку:

  ```
  <Ctrl k>
  ```

- Искать в файле по регулярному выражению (используйте `<Ctrl n>`/`<Ctrl p>` чтобы перейти к следующему/предыдущему совпадению):

  ```
  <Ctrl f>{{шаблон}}<Ввод>
  ```

- Выполнить команду:

  ```
  <Ctrl e>{{команда}}<Ввод>
  ```

- Выполнить замену во всем файле:

  ```
  <Ctrl e>replaceall "{{шаблон}}" "{{замена}}"<Ввод>
  ```

- Выход:

  ```
  <Ctrl q>
  ```

# mkdir

Создание директорий и установка их прав доступа.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/mkdir-invocation.html.

- Создать указанные директории:

```
mkdir {{путь/к/каталогу1 путь/к/каталогу2 ...}}
```

- Создать указанные директории, включая родительские, при необходимости:

```
mkdir {{[-p|--parents]}} {{путь/к/каталогу1 путь/к/
каталогу2 ...}}
```

- Создать директории с указанными правами доступа:

```
mkdir {{[-m|--mode]}} {{rwxrw-r--}} {{путь/к/каталогу1 путь/
к/каталогу2 ...}}
```

- Создать несколько вложенных директорий рекурсивно:

```
mkdir {{[-p|--parents]}} {{path/to/{a,b}/{x,y,z}/{h,i,j}}}
```

# mscore

Эта команда — псевдоним для **`musescore`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr musescore
  ```

# netstat

Отображать сетевую информацию, такую как активные соединения, открытые порты сокетов и т.д.

Смотрите также: **ss**.

Больше информации: https://manned.org/netstat.

- Показать все порты:

```
netstat {{[-a|--all]}}
```

- Показать все прослушиваемые порты:

```
netstat {{[-l|--listening]}}
```

- Показать прослушиваемые TCP-порты:

```
netstat {{[-t|--tcp]}}
```

- Показать идентификатор процесса (PID) и имена программ:

```
netstat {{[-p|--program]}}
```

- Выводить информацию непрерывно:

```
netstat {{[-c|--continuous]}}
```

- Показать таблицу маршрутизации и не преобразовывать IP-адреса в имена хостов:

```
netstat {{[-rn|--route --numeric]}}
```

- Показать прослушиваемые TCP и UDP порты (с указанием пользователя и процесса, если запущено от имени root):

```
netstat {{[-tulpne|--tcp --udp --listening --program --
numeric --extend]}}
```

# nohup

Позволяет процессу продолжать работу после закрытия терминала.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/nohup-invocation.html.

- Запустить процесс, который может выполняться в отвязке от терминала:

```
nohup {{команда}} {{аргумент1 аргумент2 ...}}
```

- Запустить `nohup` в фоновом режиме:

```
nohup {{команда}} {{аргумент1 аргумент2 ...}} &
```

- Запустить скрипт оболочки, который может выполняться в отвязке от терминала:

```
nohup {{путь/до/скрипта.sh}} &
```

- Запустить процесс и перенаправить его вывод в указанный файл:

```
nohup {{команда}} {{аргумент1 аргумент2 ...}} > {{путь/до/выходного_файла}} &
```

# ping

Отправка пакетов ICMP ECHO_REQUEST сетевым узлам.

Больше информации: https://manned.org/ping.

- Пинговать хост:

```
ping {{хост}}
```

- Пинговать хост определенное количество раз:

```
ping -c {{количество}} {{хост}}
```

- Пинговать хост с заданным интервалом в секундах между запросами:

```
ping -i {{секунды}} {{хост}}
```

- Пинговать хост без попытки определить символьные имена для адресов:

```
ping -n {{хост}}
```

- Пинговать хост со звуковым сигналом при получении ответа (полезно для проверки соединения вручную):

```
ping -a {{хост}}
```

- Также отображать сообщение, если ответ не был получен:

```
ping -O {{хост}}
```

- Пинговать хост с определенным количеством запросов, таймаутом для каждого пакета и общим таймаутом выполнения:

```
ping -c {{количество}} -W {{секунды}} -w {{секунды}} {{хост}}
```

# pio init

Эта команда — псевдоним для **`pio project`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr pio project
  ```

# piodebuggdb

Эта команда — псевдоним для **`pio debug`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr pio debug
  ```

# platformio

Эта команда — псевдоним для **`pio`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr pio
  ```

# pwd

Вывести название текущего/рабочего каталога.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/pwd-invocation.html.

- Вывести текущий каталог:

```
pwd
```

- Вывести текущий каталог, разрешая все символические ссылки (т.е. показать "физический" путь):

```
pwd {{[-P|--physical]}}
```

- Отобразить помощь:

```
pwd --help
```

# r2

Эта команда — псевдоним для **radare2**.

- Смотри документацию для оригинальной команды:

```
tldr radare2
```

# rm

Удалить файлы или каталоги.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/rm-invocation.html.

- Удалить файлы из определённых мест:

```
rm {{путь/до/файла1 путь/до/файла2 ...}}
```

- Интерактивное удаление нескольких файлов с запросом перед каждым удалением:

```
rm -i {{путь/до/файла1 путь/до/файла2 ...}}
```

- Удаление файлов с подробным выводом, печать сообщения для каждого удаленного файла:

```
rm -v {{путь/до/директории/*}}
```

- Рекурсивно удалить каталог и все его подкаталоги:

```
rm -r {{путь/до/директории}}
```

# ssh

Secure Shell (SSH) — это протокол для безопасного входа на удаленные системы.

Его можно использовать для входа или выполнения команд на удаленном сервере.

Больше информации: https://man.openbsd.org/ssh.

- Подключиться к удаленному серверу:

```
ssh {{имя_пользователя}}@{{удаленный_хост}}
```

- Подключиться к удаленному серверу с использованием определенного приватного ключа:

```
ssh -i {{путь/к/файлу_ключа}} {{имя_пользователя}}
@{{удаленный_хост}}
```

- Подключиться к удаленному серверу с IP-адресом 10.0.0.1, используя определенный порт [p]:

```
ssh {{имя_пользователя}}@10.0.0.1 -p {{2222}}
```

- Запустить команду на удаленном сервере с выделением [t]ty, что позволяет взаимодействовать с удаленной командой:

```
ssh {{имя_пользователя}}@{{удаленный_хост}} -t {{команда}}
{{аргументы_команды}}
```

- SSH-туннелирование: динамическая [D] переадресация портов (SOCKS-прокси на `localhost:1080`):

```
ssh -D {{1080}} {{имя_пользователя}}@{{удаленный_хост}}
```

- SSH-туннелирование: переадресовать определенный порт (`localhost:9999` на `example.com:80`) с отключением псевдо-[T]ty и выполне[N]ия удаленных команд:

```
ssh -L {{9999}}:{{example.com}}:{{80}} -N -T
{{имя_пользователя}}@{{удаленный_хост}}
```

- SSH-прыжок [J]: подключиться к удаленному серверу через промежуточный хост (можно указать несколько хостов через запятую):

```
ssh -J {{имя_пользователя}}@{{промежуточный_хост}}
{{имя_пользователя}}@{{удаленный_хост}}
```

- Закрыть зависшую сессию:

```
<Enter><~><.>
```

# stat

Показ информации о файле и файловой системе.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/stat-invocation.html.

- Показать свойства файла, такие как размер, права доступа, даты создания и последнего обращения и другие:

```
stat {{путь/к/файлу}}
```

- То же, что и выше, но в сжатой форме:

```
stat {{[-t|--terse]}} {{путь/к/файлу}}
```

- Показать информацию о файловой системе, на которой расположен указанный файл:

```
stat {{[-f|--file-system]}} {{путь/к/файлу}}
```

- Показать только права доступа в восьмеричном виде:

```
stat {{[-c|--format]}} "%a %n" {{путь/к/файлу}}
```

- Показать владельца и группу файла:

```
stat {{[-c|--format]}} "%U %G" {{путь/к/файлу}}
```

- Показать размер файла в байтах:

```
stat {{[-c|--format]}} "%s %n" {{путь/к/файлу}}
```

# stty

Настройка параметров интерфейса терминального устройства.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/stty-invocation.html.

- Показать размеры текущего терминала:

```
stty size
```

- Показать все настройки текущего терминала:

```
stty {{[-a|--all]}}
```

- Задать количество строк или столбцов:

```
stty {{rows|cols}} {{количество}}
```

- Получить фактическую скорость передачи данных устройства:

```
stty {{[-F|--file]}} {{путь/к/файлу_устройства}} speed
```

- Сбросить все режимы до разумных значений для текущего терминала:

```
stty sane
```

- Переключиться между сырым (необработанным) и нормальным режимами:

```
stty {{raw|cooked}}
```

- Отключить или включить отображение вводимых символов:

```
stty {{-echo|echo}}
```

- Показать справку:

```
stty --help
```

# sudo

Выполнять команду от имени суперпользователя или другого пользователя.

Больше информации: https://www.sudo.ws/sudo.html.

- Выполнить команду от имени суперпользователя:

```
sudo {{less /var/log/syslog}}
```

- Отредактировать файл от имени суперпользователя, используя редактор по умолчанию:

```
sudo {{[-e|--edit]}} {{/etc/fstab}}
```

- Выполнить команду от имени другого пользователя и/или группы:

```
sudo {{[-u|--user]}} {{пользователь}} {{[-g|--group]}}
{{группа}} {{id -a}}
```

- Повторить последнюю команду, добавив перед ней `sudo` (работает в Bash, Zsh и т.д.):

```
sudo !!
```

- Запустить оболочку по умолчанию с правами суперпользователя, выполнив файлы входа в систему (`.profile`, `.bash_profile` и т.д.):

```
sudo {{[-i|--login]}}
```

- Запустить оболочку по умолчанию с правами суперпользователя, не меняя переменных окружения:

```
sudo {{[-s|--shell]}}
```

- Запустить оболочку по умолчанию от имени указанного пользователя, загрузив его окружение и файлы входа в систему (`.profile`, `.bash_profile` и т.д.):

```
sudo {{[-i|--login]}} {{[-u|--user]}} {{пользователь}}
```

- Показать список разрешенных (и запрещенных) команд для текущего пользователя:

```
sudo {{[-ll|--list --list]}}
```

# tar

Утилита архивирования.

Обычно используется в сочетании с методом сжатия, таким как **`gzip`** или **`bzip2`**.

Больше информации: https://www.gnu.org/software/tar/manual/tar.html.

- Создать ([c]reate) архив и записать его в файл ([f]ile):

```
tar cf {{путь/к/целевому.tar}} {{путь/к/файлу1 путь/к/файлу2 ...}}
```

- Создать ([c]reate) сжатый в g[z]ip архив и записать его в файл ([f]ile):

```
tar czf {{путь/к/целевому.tar.gz}} {{путь/к/файлу1 путь/к/файлу2 ...}}
```

- Создать ([c]reate) сжатый в g[z]ip архив из каталога, используя относительные пути:

```
tar czf {{путь/к/целевому.tar.gz}} {{[-C|--directory]}} {{путь/к/каталогу}} .
```

- Извлечь (e[x]tract) (сжатый) файл ([f]ile) архива в текущий каталог с подробным ([v]erbosely) выводом:

```
tar xvf {{путь/к/исходному.tar[.gz|.bz2|.xz]}}
```

- Извлечь (e[x]tract) (сжатый) файл ([f]ile) архива в указанный каталог:

```
tar xf {{путь/к/исходному.tar[.gz|.bz2|.xz]}} {{[-C|--directory]}} {{путь/к/каталогу}}
```

- Создать ([c]reate) сжатый архив и записать его в файл ([f]ile), используя расширение файла для автоматического ([a]utomatically) определения программы сжатия:

```
tar caf {{путь/к/целевому.tar.xz}} {{путь/к/файлу1 путь/к/файлу2 ...}}
```

- Вывести список (lis[t]) содержимого tar-файла ([f]ile) с подробным ([v]erbosely) выводом:

```
tar tvf {{путь/к/исходному.tar}}
```

- Извлечь (e[x]tract) файлы, соответствующие шаблону, из файла ([f]ile) архива:

```
tar xf {{путь/к/исходному.tar}} --wildcards "{{*.html}}"
```

# tldr

Показывает простые страницы помощи для инструментов коммандной строки из проекта tldr-pages.

Примечание: параметры **`--language`** и **`--list`** не требуются в спецификации клиента, но большинство клиентов их используют.

Больше информации: https://github.com/tldr-pages/tldr/blob/main/CLIENT-SPECIFICATION.md#command-line-interface.

- Вывести страницу tldr для конкретной команды (подсказка: именно так вы сюда попали!):

```
tldr {{команда}}
```

- Вывести страницу tldr для конкретной подкоманды:

```
tldr {{команда}} {{подкоманда}}
```

- Вывести страницу tldr для команды на указанном языке (если доступно, иначе используется английский):

```
tldr {{[-L|--language]}} {{код_языка}} {{команда}}
```

- Вывести страницу tldr для команды с конкретной платформы:

```
tldr {{[-p|--platform]}} {{android|common|freebsd|linux|osx|
netbsd|openbsd|sunos|windows}} {{команда}}
```

- Обновить локальный кэш страниц tldr:

```
tldr {{[-u|--update]}}
```

- Вывести список всех страниц для текущей платформы и категории `common`:

```
tldr {{[-l|--list]}}
```

- Вывести список всех доступных страниц подкоманд для команды:

```
tldr {{[-l|--list]}} | grep {{команда}} | column
```

- Вывести страницу tldr для случайной команды:

```
tldr {{[-l|--list]}} | shuf {{[-n|--head-count]}} 1 | xargs
tldr
```

# tldrl

Эта команда — псевдоним для **`tldr-lint`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr tldr-lint
  ```

# tlmgr arch

Эта команда — псевдоним для **`tlmgr platform`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr tlmgr platform
  ```

# tput

Просмотр и изменение настроек и возможностей терминала.

Больше информации: https://manned.org/tput.

- Переместить курсор в определённое место на экране:

```
tput cup {{строка}} {{столбец}}
```

- Установить цвет переднего плана (af) или фона (ab):

```
tput {{setaf|setab}} {{ansi_код_цвета}}
```

- Обратить цвета текста и фона:

```
tput rev
```

- Сбросить все атрибуты текста терминала:

```
tput sgr0
```

- Показать количество столбцов, строк или цветов:

```
tput {{cols|lines|colors}}
```

- Включить или отключить перенос слов:

```
tput {{smam|rmam}}
```

- Спрятать или показать курсор терминала:

```
tput {{civis|cnorm}}
```

- Сохранить или восстановить состояние текста терминала (smcup также перехватывает события колеса прокрутки):

```
tput {{smcup|rmcup}}
```

# tty

Выводит название терминала.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/tty-invocation.html.

- Вывести имя файла, соответствующее текущему терминалу:

```
tty
```

# uname

Показать информацию о текущей машине и операционной системе.

Смотрите также: **`lsb_release`**.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/uname-invocation.html.

- Показать имя ядра:

```
uname
```

- Показать всю доступную системную информацию:

```
uname {{[-a|--all]}}
```

- Показать архитектуру системы и информацию о процессоре:

```
uname {{[-mp|--machine --processor]}}
```

- Показать имя ядра, релиз ядра и версию ядра:

```
uname {{[-srv|--kernel-name --kernel-release --kernel-version]}}
```

- Показать сетевое имя хоста системы:

```
uname {{[-n|--nodename]}}
```

- Показать имя текущей операционной системы:

```
uname {{[-o|--operating-system]}}
```

- Показать справку:

```
uname --help
```

# unzip

Извлекает сжатые файлы из архива zip.

Больше информации: https://manned.org/unzip.

- Распаковать файл(ы) zip (для нескольких файлов укажите пути через пробел):

```
unzip {{архив(ы)}}
```

- Распаковать файл(ы) по нужному пути:

```
unzip {{архив(ы)}} -d {{/путь/куда/положить/
извлечённый_файл(ы)}}
```

- Вывести список файлов в архиве zip, не распаковывая их:

```
unzip -l {{архив.zip}}
```

- Извлечь содержимое файла в `stdout` вместе с именами распакованных файлов:

```
unzip -c {{архив.zip}}
```

- Распаковать архив zip, который был создан на windows и содержит не-ascii имена файлов (напр. кириллица):

```
unzip -O {{gbk}} {{архив.zip}}
```

# vi

Эта команда — псевдоним для **`vim`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr vim
  ```

# weasyprint

Перевод HTML в PDF или PNG.

Больше информации: https://doc.courtbouillon.org/weasyprint/stable/api_reference.html#command-line-api.

- Перевести HTML файл в PDF:

```
weasyprint {{путь/ко/входному.html}} {{путь/к/выходному.pdf}}
```

- Перевести HTML файл в PNG, включая дополнительные пользовательские таблицы стилей:

```
weasyprint {{путь/ко/входному.html}} {{путь/к/выходному.png}}
--stylesheet {{путь/к/таблице_стилей.css}}
```

- При переводе выводить дополнительную отладочную информацию:

```
weasyprint {{путь/ко/входному.html}} {{путь/к/выходному.pdf}}
--verbose
```

- При выводе в PNG указать нестандартное разрешение:

```
weasyprint {{путь/ко/входному.html}} {{путь/к/выходному.png}}
--resolution {{300}}
```

- Во входном HTML файле указать базовый URL для относительных URLs:

```
weasyprint {{путь/ко/входному.html}} {{путь/к/выходному.png}}
--base-url {{url_или_имя_файла}}
```

# which

Отобразить абсолютный путь к программе.

Больше информации: https://manned.org/which.

- Найти переменную окружения PATH и отобразить расположение всех соответствующих исполняемых файлов:

```
which {{исполняемый_файл}}
```

- Если есть несколько исполняемых файлов, которые совпадают, отобразить все:

```
which {{[-a|--all]}} {{исполняемый_файл}}
```

# zip

Упаковывает и сжимает (архивирует) файлы в файл zip.

Смотрите также: **`unzip`**.

Больше информации: https://manned.org/zip.

- Добавить файлы/папки в указанный архив ([r]ecursively):

```
zip {{[-r|--recurse-paths]}} {{путь/до/архива.zip}} {{путь/до/файла_или_папки1 путь/до/файла_или_папки2 ...}}
```

- Удалить файлы/папки из указанного архива ([d]elete):

```
zip {{[-d|--delete]}} {{путь/до/архива.zip}} {{путь/до/файла_или_папки1 путь/до/файла_или_папки2 ...}}
```

- Заархивировать файлы/папки, исключая некоторые (e[x]clude):

```
zip {{[-r|--recurse-paths]}} {{путь/до/архива.zip}} {{путь/до/файла_или_папки1 путь/до/файла_или_папки2 ...}} {{[-x|--exclude]}} {{путь/до/исключаемых_файлов_или_папок}}
```

- Заархивировать файлы/папки с заданной степенью сжатия (`0` — без сжатия, `9` — максимальная):

```
zip {{[-r|--recurse-paths]}} -{{0-9}} {{путь/до/архива.zip}} {{путь/до/файла_или_папки1 путь/до/файла_или_папки2 ...}}
```

- Создать зашифрованный паролем архив ([e]ncrypted):

```
zip {{[-re|--recurse-paths --encrypt]}} {{путь/до/архива.zip}} {{путь/до/файла_или_папки1 путь/до/файла_или_папки2 ...}}
```

- Заархивировать файлы/папки в многотомный архив ([s]plit), например, частями по 3 Гб:

```
zip {{[-rs|--recurse-paths --split-size]}} {{3g}} {{путь/до/архива.zip}} {{путь/до/файла_или_папки1 путь/до/файла_или_папки2 ...}}
```

- Вывести содержимое указанного архива ([s]how [f]iles):

```
zip {{[-sf|--split-size --freshen]}} {{путь/до/архива.zip}}
```

# zsh

Z SHell — командный интерпретатор, совместимый с Bash.

Смотри также **`histexpand`** про подстановку команд из списка истории.

Больше информации: https://zsh.sourceforge.io/Doc/Release/Invocation.html#Invocation.

- Запустить интерактивную сессию оболочки:

  ```
  zsh
  ```

- Выполнить команду и выйти:

  ```
  zsh -c "{{команда}}"
  ```

- Выполнить скрипт:

  ```
  zsh {{путь/до/скрипта.zsh}}
  ```

- Выполнить скрипт с выводом каждой команды перед её выполнением:

  ```
  zsh --xtrace {{путь/до/скрипта.zsh}}
  ```

- Запустить интерактивную сессию оболочки в подробном режиме, выводя каждую команду перед её выполнением:

  ```
  zsh --verbose
  ```

- Выполнить определённую команду внутри Zsh с отключёнными glob-шаблонами:

  ```
  noglob {{команда}}
  ```

# Linux

# add-apt-repository

Управление определениями репозиториев **`apt`**.

Больше информации: https://manned.org/add-apt-repository.

- Добавить новый репозиторий `apt`:

```
add-apt-repository {{спецификация_репозитория}}
```

- Удалить репозиторий `apt`:

```
add-apt-repository {{[-r|--remove]}}
{{спецификация_репозитория}}
```

- Обновить кэш пакетов после добавления репозитория:

```
add-apt-repository --update {{спецификация_репозитория}}
```

- Разрешить загрузку исходных пакетов из репозитория:

```
add-apt-repository {{[-s|--enable-source]}}
{{спецификация_репозитория}}
```

# apk

Инструмент управления пакетами Alpine Linux.

Больше информации: https://wiki.alpinelinux.org/wiki/Alpine_Package_Keeper.

- Обновить индекс репозиториев и обновить все пакеты:

```
apk upgrade {{[-U|--update-cache]}}
```

- Обновить только индекс репозиториев:

```
apk update
```

- Установить новый пакет:

```
apk add {{пакет}}
```

- Удалить пакет:

```
apk del {{пакет}}
```

- Отремонтировать/переустановить пакет без изменения основных зависимостей:

```
apk fix {{пакет}}
```

- Искать пакеты по ключевому слову и вывести результаты с их описаниями:

```
apk search {{[-v|--verbose]}} {{ключевое_слово}}
```

- Искать пакеты по ключевому слову в их описании:

```
apk search {{[-d|--description]}} {{ключевое_слово}}
```

- Отобразить информацию о конкретном пакете:

```
apk info {{пакет}}
```

# apt-add-repository

Эта команда - псевдоним команды **`add-apt-repository`**.

- Отобразить документацию по исходной команде:

  ```
  tldr add-apt-repository
  ```

# apt-cache

Инструмент поиска пакетов для Debian и Ubuntu.

Больше информации: https://manned.org/apt-cache.8.

- Поиск пакета в локальной базе пакетов:

```
apt-cache search {{запрос}}
```

- Показать информацию о пакете:

```
apt-cache show {{пакет}}
```

- Показать, установлен ли пакет и обновлен ли он:

```
apt-cache policy {{пакет}}
```

- Показать зависимости для пакета:

```
apt-cache depends {{пакет}}
```

- Показать пакеты, которые зависят от определенного пакета:

```
apt-cache rdepends {{пакет}}
```

# apt-file

Поиск файлов в пакетах **`apt`**, в том числе еще не установленных.

Больше информации: https://manned.org/apt-file.1.

- Обновить базу данных метаданных:

```
sudo apt update
```

- Поиск пакетов, содержащих указанный файл или путь:

```
apt-file {{search|find}} {{частичный_путь/к/файлу}}
```

- Вывести содержимое конкретного пакета:

```
apt-file {{show|list}} {{пакет}}
```

- Поиск пакетов, соответствующих регулярному выражению:

```
apt-file {{search|find}} {{[-x|--regexp]}}
{{регулярное_выражение}}
```

# apt-get

Утилита управления пакетами для Debian и Ubuntu.

Для поиска пакетов используйте **`apt-cache`**.

Рекомендуется использовать **`apt`** при интерактивном использовании в Ubuntu начиная с версии 16.04.

Больше информации: https://manned.org/apt-get.8.

- Обновить список доступных пакетов и их версий (рекомендуется перед выполнением других команд `apt-get`):

  ```
  apt-get update
  ```

- Установить пакет или обновить его до последней версии:

  ```
  apt-get install {{пакет}}
  ```

- Удалить пакет:

  ```
  apt-get remove {{пакет}}
  ```

- Удалить пакет и его конфигурационные файлы:

  ```
  apt-get purge {{пакет}}
  ```

- Обновить все установленные пакеты до последних версий:

  ```
  apt-get upgrade
  ```

- Очистить локальный репозиторий - удалить файлы пакетов (`.deb`) из прерванных загрузок, которые больше не могут быть загружены:

  ```
  apt-get autoclean
  ```

- Удалить все пакеты, которые больше не нужны, так как пакеты, от которых они зависели, были удалены:

  ```
  apt-get autoremove
  ```

- Обновить установленные пакеты (как с помощью команды `upgrade`), но удалить устаревшие пакеты и установить дополнительные пакеты, чтобы удовлетворить новые зависимости:

  ```
  apt-get dist-upgrade
  ```

# apt install

Установить пакеты для дистрибутивов на базе Debian.

Больше информации: https://manned.org/apt.8.

- Установить пакет или обновить его до последней версии:

```
sudo apt install {{пакет}}
```

- Отобразить подробную информацию о версии пакета во время установки или обновления:

```
sudo apt install {{[-V|--verbose-versions]}} {{пакет}}
```

# apt-key

Утилита управления ключами для диспетчера пакетов APT в Debian и Ubuntu.

Примечание: **`apt-key`** теперь устарел (за исключением использования **`apt-key del`** в скриптах разработчиков).

Больше информации: https://manned.org/apt-key.8.

- Список доверенных ключей:

```
apt-key list
```

- Добавить ключ в список доверенных ключей:

```
apt-key add {{файл_публичного_ключа.asc}}
```

- Удалить ключ из списка доверенных ключей:

```
apt-key del {{идентификатор_ключа}}
```

- Добавить удалённый ключ в список доверенных ключей:

```
wget {{[-qO|--quiet --output-document]}} - {{https://
host.tld/файл.key}} | apt-key add -
```

- Добавить ключ с сервера ключей, указав только идентификатор ключа:

```
apt-key adv --keyserver {{pgp.mit.edu}} --recv
{{идентификатор_ключа}}
```

# apt-mark

Утилита для изменения статуса установленных пакетов.

Больше информации: https://manned.org/apt-mark.8.

- Отметить пакет как автоматически установленный:

```
sudo apt-mark auto {{пакет}}
```

- Удерживать пакет в его текущей версии и запретить его обновление:

```
sudo apt-mark hold {{пакет}}
```

- Перестать удерживать пакет и разрешить его обновление:

```
sudo apt-mark unhold {{пакет}}
```

- Показать пакеты, установленные вручную:

```
apt-mark showmanual
```

- Показать удерживаемые пакеты:

```
apt-mark showhold
```

# apt moo

Пасхальное яйцо **APT**.

Больше информации: https://manned.org/apt.8.

- Отобразить пасхальное яйцо с изображением коровы:

```
apt moo
```

# apt

Менеджер пакетов для дистрибутивов на базе Debian.

Предназначен как удобная альтернатива **`apt-get`** для интерактивного использования.

Эквивалентные команды в других пакетных менеджерах смотрите в https://wiki.archlinux.org/title/Pacman/Rosetta.

Больше информации: https://manned.org/apt.8.

- Обновить список доступных пакетов и их версий (рекомендуется перед выполнением других команд `apt`):

  ```
  sudo apt update
  ```

- Поиск пакетов по названию или описанию:

  ```
  apt search {{пакет}}
  ```

- Поиск пакетов только по имени (поддерживает такие символы подстановки, как `*`):

  ```
  apt list {{пакет}}
  ```

- Показать подробную информацию о пакете:

  ```
  apt show {{пакет}}
  ```

- Установить пакет или обновить его до последней версии:

  ```
  sudo apt install {{пакет}}
  ```

- Удалить пакет (используйте `purge`, чтобы удалить и конфигурационные файлы):

  ```
  sudo apt remove {{пакет}}
  ```

- Обновить все установленные пакеты до последних версий:

  ```
  sudo apt upgrade
  ```

- Вывести список всех установленных пакетов:

  ```
  apt list {{[-i|--installed]}}
  ```

# ark

Утилита для архивации от KDE.

Больше информации: https://docs.kde.org/stable5/en/ark/ark/.

- Извлечь указанный архив в текущий каталог:

```
ark {{[-b|--batch]}} {{путь/к/архиву}}
```

- Извлечь архив в указанный каталог:

```
ark {{[-b|--batch]}} {{[-o|--destination]}} {{путь/к/
каталогу}} {{путь/к/архиву}}
```

- Создать архив, если он не существует, и добавить в него указанные файлы:

```
ark {{[-t|--add-to]}} {{путь/к/архиву}} {{путь/к/файлу1 путь/
к/файлу2 ...}}
```

# cal

Отображать календарь с выделением текущего дня.

Больше информации: https://manned.org/cal.

- Показать календарь на текущий месяц:

```
cal
```

- Показать 3 месяца (предыдущий, текущий и следующий):

```
cal {{[-3|--three]}}
```

- Показать календарь на весь текущий год:

```
cal {{[-y|--year]}}
```

- Показать следующие двенадцать месяцев:

```
cal {{[-Y|--twelve]}}
```

- Использовать понедельник как первый день недели:

```
cal {{[-m|--monday]}}
```

- Показать календарь на указанный год (4 цифры):

```
cal {{год}}
```

- Показать календарь на указанные месяц и год:

```
cal {{месяц}} {{год}}
```

# cat

Вывести и объединить файлы.

Больше информации: https://www.gnu.org/software/coreutils/manual/html_node/cat-invocation.html.

- Вывести содержимое файла в `stdout`:

```
cat {{путь/к/файлу}}
```

- Объединить несколько файлов в один выходной файл:

```
cat {{путь/к/файлу1 путь/к/файлу2 ...}} > {{путь/к/выходному_файлу}}
```

- Добавить содержимое нескольких файлов в конец выходного файла:

```
cat {{путь/к/файлу1 путь/к/файлу2 ...}} >> {{путь/к/выходному_файлу}}
```

- Записать `stdin` в файл:

```
cat - > {{путь/к/файлу}}
```

- Пронумеровать все строки вывода:

```
cat {{[-n|--number]}} {{путь/к/файлу}}
```

- Отобразить непечатаемые и пробельные символы (с префиксом `M-` для не-ASCII символов):

```
cat {{[-vte|--show-nonprinting -t -e]}} {{путь/к/файлу}}
```

# crond

Демон для выполнения запланированных команд из файлов crontab.

Больше информации: https://manned.org/crond.

- Запустить демона в фоновом режиме и проверять наличие запланированных команд:

  ```
  crond
  ```

- Запустить демона на переднем плане и проверять наличие запланированных команд:

  ```
  crond -n
  ```

- Отправлять вывод заданий от демона в [s]истемный журнал:

  ```
  crond -s
  ```

- Игнорировать ограничения по умолчанию и принимать пользовательские файлы crontab:

  ```
  crond -p
  ```

- Наследовать путь к файлу crontab из переменных окружения:

  ```
  crond -P
  ```

# darling

Запускать программы для macOS в Linux.

Больше информации: https://darlinghq.org.

- Запустить встроенную команду:

```
darling shell {{uname}}
```

- Запустить указанную программу с аргументами:

```
darling shell {{путь/к/программе}} {{аргумент_программы_1
аргумент_программы_2 ...}}
```

- Открыть оболочку macOS:

```
darling shell
```

- Завершить работу сервиса:

```
darling shutdown
```

# dnf download

Загрузка RPM-пакетов из репозиториев DNF.

Не является частью основной команды **`dnf`**, но поддерживается через плагин **`dnf-plugins-core`**.

Смотрите также: **`dnf`**.

Больше информации: https://dnf-plugins-core.readthedocs.io/en/latest/download.html.

- Скачать последнюю версию пакета в текущую директорию:

  ```
  dnf download {{пакет}}
  ```

- Скачать пакет в указанную директорию (папка должна существовать):

  ```
  dnf download {{пакет}} --destdir {{путь/к/директории}}
  ```

- Вывести URL, откуда можно скачать RPM-пакет:

  ```
  dnf download --url {{пакет}}
  ```

# dnf group

Управление виртуальными коллекциями пакетов в системах на базе Fedora.

Больше информации: https://dnf.readthedocs.io/en/latest/command_ref.html#group-command.

- Показать список групп DNF с указанием статуса установки в таблице:

```
dnf {{[grp|group]}} list
```

- Показать информацию о группе DNF, включая репозиторий и опциональные пакеты:

```
dnf {{[grp|group]}} info {{имя_группы}}
```

- Установить группу DNF:

```
dnf {{[grp|group]}} install {{имя_группы}}
```

- Удалить группу DNF:

```
dnf {{[grp|group]}} remove {{имя_группы}}
```

- Обновить группу DNF:

```
dnf {{[grp|group]}} upgrade {{имя_группы}}
```

# dnf install

Установка пакетов в дистрибутивах на базе Red Hat.

Больше информации: https://dnf.readthedocs.io/en/latest/command_ref.html#install-examples.

- Установить пакеты по имени:

```
sudo dnf {{[in|install]}} {{пакет1 пакет2 ...}}
```

- Установить пакет из локального файла:

```
sudo dnf {{[in|install]}} {{путь/к/файлу}}
```

- Установить пакет из интернета:

```
sudo dnf {{[in|install]}} {{https://example.com/package.rpm}}
```

- Добавить репозитории Extra Packages for Enterprise Linux (EPEL):

```
sudo dnf {{[in|install]}} https://dl.fedoraproject.org/pub/epel/epel-release-latest-{{10}}.noarch.rpm
```

- Добавить RPM-репозиторий Remi:

```
sudo dnf {{[in|install]}} https://rpms.remirepo.net/enterprise/remi-release-{{8}}.rpm
```

# dnf module

Управление модульностью пакетов.

Больше информации: https://dnf.readthedocs.io/en/latest/command_ref.html#module-command.

- Просмотр общего обзора модульности:

  ```
  dnf module list
  ```

- Просмотр модульности конкретной программы:

  ```
  dnf module list {{имя_пакета}}
  ```

- Включить пакет:

  ```
  sudo dnf module enable {{имя_пакета}}:{{поток}}
  ```

- Включить и установить конкретную версию:

  ```
  dnf module install {{имя_пакета}}:{{поток}}
  ```

# dnf repoquery

Запрос информации о пакетах.

Больше информации: https://dnf.readthedocs.io/en/latest/command_ref.html#repoquery-command.

- Запросить зависимости пакета:

```
dnf {{[rq|repoquery]}} --deplist {{пакет}}
```

# dnf

Утилита управления пакетами для RHEL, Fedora и CentOS (заменяет yum).

Некоторые подкоманды, такие как **`group`** и **`config-manager`**, имеют свою собственную документацию по использованию.

Эквивалентные команды в других пакетных менеджерах смотрите в [https://wiki.archlinux.org/title/Pacman/Rosetta](https://wiki.archlinux.org/title/Pacman/Rosetta).

Больше информации: [https://dnf.readthedocs.io](https://dnf.readthedocs.io).

- Обновить установленные пакеты до новейших доступных версий:

  ```
  sudo dnf {{[up|upgrade]}}
  ```

- Поиск пакетов по ключевым словам:

  ```
  dnf {{[se|search]}} {{ключевое_слово1 ключевое_слово2 ...}}
  ```

- Отобразить подробную информацию о пакете:

  ```
  dnf {{[if|info]}} {{пакет}}
  ```

- Установить новый пакет (используйте --assumeyes для автоматического подтверждения всех запросов):

  ```
  sudo dnf {{[in|install]}} {{пакет1 пакет2 ...}}
  ```

- Удалить пакет:

  ```
  sudo dnf {{[rm|remove]}} {{пакет1 пакет2 ...}}
  ```

- Список установленных пакетов:

  ```
  dnf {{[ls|list]}} --installed
  ```

- Найти какие пакеты предоставляют данную команду:

  ```
  dnf {{[wp|provides]}} {{команда}}
  ```

- Просмотр всех прошлых операций:

  ```
  dnf {{[hist|history]}}
  ```

# free

Показать количество свободной и используемой памяти в системе.

Больше информации: https://manned.org/free.

- Показать системную память:

```
free
```

- Показать память в байтах/килобайтах/мегабайтах/гигабайтах:

```
free -{{b|k|m|g}}
```

- Показать память в человеко-читаемом формате:

```
free {{[-h|--human]}}
```

- Обновлять вывод каждые 2 секунды:

```
free {{[-s|--seconds]}} 2
```

# hexdump

Дамп файла в ASCII, десятичном, шестнадцатеричном и восьмеричном форматах.

Больше информации: https://manned.org/hexdump.

- Распечатать шестнадцатеричное представление файла, заменяя повторяющиеся строки на '*':

```
hexdump {{путь/до/файла}}
```

- Отобразить шестнадцатеричное и ASCII представление в две колонки:

```
hexdump {{[-C|--canonical]}} {{путь/до/файла}}
```

- Отобразить двухколончатое представление файла, обработав только указанное число байтов с начала:

```
hexdump {{[-C|--canonical]}} {{[-n|--length]}}
{{количество_байтов}} {{путь/до/файла}}
```

- Не заменять повторяющиеся строки на '*':

```
hexdump {{[-v|--no-squeezing]}} {{путь/до/файла}}
```

# hostnamectl

Получение и установка имени хоста компьютера.

Больше информации: https://www.freedesktop.org/software/systemd/man/hostnamectl.html.

- Получить имя хоста компьютера:

```
hostnamectl
```

- Задать имя хоста компьютера:

```
sudo hostnamectl set-hostname "{{имя_хоста}}"
```

- Задать красивое (короткое) имя хоста компьютера:

```
sudo hostnamectl set-hostname --static
"{{имя_хоста.example.com}}" && sudo hostnamectl set-hostname
--pretty "{{имя_хоста}}"
```

- Сбросить имя хоста компьютера к значению по умолчанию:

```
sudo hostnamectl set-hostname --pretty ""
```

# ip route list

Эта команда — псевдоним для **`ip route show`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr ip route show
  ```

# ip

Показывать и управлять маршрутизацией, устройствами, политиками маршрутизации и туннелями.

Некоторые подкоманды, такие как **`address`**, имеют собственную документацию.

Больше информации: https://manned.org/ip.8.

- Показать детальную информацию об интерфейсах:

```
ip {{[a|address]}}
```

- Показать краткую информацию о сетевом уровне для интерфейсов:

```
ip {{[-br|-brief]}} {{[a|address]}}
```

- Показать краткую информацию о канальном уровне для интерфейсов:

```
ip {{[-br|-brief]}} {{[l|link]}}
```

- Показать таблицу маршрутизации:

```
ip {{[r|route]}}
```

- Показать соседей (ARP-таблицу):

```
ip {{[n|neighbour]}}
```

- Включить/выключить интерфейс:

```
sudo ip {{[l|link]}} {{[s|set]}} {{интерфейс}} {{up|down}}
```

- Добавить/Удалить IP-адрес для интерфейса:

```
sudo ip {{[a|address]}} {{add|delete}} {{ip_адрес}}/{{маска}}
dev {{интерфейс}}
```

- Добавить маршрут по умолчанию:

```
sudo ip {{[r|route]}} {{[a|add]}} default via {{ip_адрес}}
dev {{интерфейс}}
```

# journalctl

Запросить записи из журнала systemd.

Больше информации: https://www.freedesktop.org/software/systemd/man/journalctl.html.

- Показать все сообщения с уровнем приоритета 3 (ошибки) с момента текущей загрузки:

```
journalctl {{[-b|--boot]}} {{[-p|--priority]}} 3
```

- Удалить записи журнала, которые старше 2 дней:

```
journalctl --vacuum-time 2d
```

- Показать только последние `n` строк и отслеживать новые сообщения (аналогично `tail -f` для традиционного syslog):

```
journalctl {{[-n|--lines]}} {{количество}} {{[-f|--follow]}}
```

- Показать все сообщения от определенного юнита:

```
journalctl {{[-u|--unit]}} {{юнит}}
```

- Показать логи для указанного юнита с момента его последнего запуска:

```
journalctl _SYSTEMD_INVOCATION_ID=$(systemctl show --value --property=InvocationID {{юнит}})
```

- Отфильтровать сообщения по временному диапазону (можно использовать метку времени или плейсхолдеры, такие как "yesterday"):

```
journalctl {{[-S|--since]}} {{now|today|yesterday|tomorrow}} {{[-U|--until]}} "{{ГГГГ-ММ-ДД ЧЧ:ММ:СС}}"
```

- Показать все сообщения от определенного процесса:

```
journalctl _PID={{id_процесса}}
```

- Показать все сообщения от определенного исполняемого файла:

```
journalctl {{путь/к/исполняемому_файлу}}
```

# lnav

Инструмент для просмотра и анализа файлов журналов (логов).

Больше информации: https://docs.lnav.org/en/latest/cli.html.

- Просмотреть логи, в качестве аргумента можно указать файл лога, каталог или URL-адрес:

  ```
  lnav {{путь/до/файла_или_директории|url-адрес}}
  ```

- Просмотреть логи удаленного хоста (требуется аутентификация SSH без пароля):

  ```
  lnav {{ssh}} {{пользователь}}@{{host1.example.com}}:{{/var/log/syslog.log}}
  ```

- Проверить файлы на соответствие корректности формату логов:

  ```
  lnav -C {{путь/до/директории_с_логами}}
  ```

# login

Начать сеанс для пользователя.

Больше информации: https://manned.org/login.

- Войти как пользователь:

  ```
  login {{пользователь}}
  ```

- Войти как пользователь без аутентификации (если он уже аутентифицирован):

  ```
  login -f {{пользователь}}
  ```

- Войти как пользователь с сохранением окружения:

  ```
  login -p {{пользователь}}
  ```

- Войти как пользователь на удаленный хост:

  ```
  login -h {{хост}} {{пользователь}}
  ```

# lsb_release

Выводит информацию, определённую стандартом LSB (Linux Standard Base), а также характерную для дистрибутива.

Больше информации: https://manned.org/lsb_release.

- Отобразить всю имеющуюся информацию:

```
lsb_release {{[-a|--all]}}
```

- Отобразить описание (обычно полное наименование) операционной системы:

```
lsb_release {{[-d|--description]}}
```

- Отобразить наименование ОС, без указания поля "Distributor ID":

```
lsb_release {{[-is|--id --short]}}
```

- Отобразить номер релиза (release number) и кодовое наименование дистрибутива без указания полей с названием:

```
lsb_release {{[-rcs|--release --codename --short]}}
```

# lsblk

Отобразить информацию об устройствах.

Больше информации: https://manned.org/lsblk.

- Отобразить список всех накопителей в древовидном виде:

```
lsblk
```

- Отобразить все устройства, в том числе "пустые":

```
lsblk {{[-a|--all]}}
```

- Отобразить столбец SIZE в байтах, а не в удобочитаемом формате:

```
lsblk {{[-b|--bytes]}}
```

- Вывод информации о файловой системе:

```
lsblk {{[-f|--fs]}}
```

- Использовать символы ASCII при отображении в формате дерева:

```
lsblk {{[-i|--ascii]}}
```

- Вывести информацию о топологии блочного устройства:

```
lsblk {{[-t|--topology]}}
```

- Исключить устройства, указанные в списке основных номеров устройств, разделенных запятыми:

```
lsblk {{[-e|--exclude]}} {{1,7,...}}
```

- Отобразить вывод с указанием списка определённых параметров, разделенных запятыми:

```
lsblk {{[-o|--output]}}
{{NAME,SERIAL,MODEL,TRAN,TYPE,SIZE,FSTYPE,MOUNTPOINT,...}}
```

# lscpu

Отображает информацию о центральном процессоре.

Больше информации: https://manned.org/lscpu.

- Отобразить информацию о центральном процессоре:

```
lscpu
```

- Отобразить информацию в виде таблицы:

```
lscpu {{[-e|--extended]}}
```

- Отобразить информацию в виде таблицы для процессорных ядер, которые находятся в отключенном состоянии:

```
lscpu {{[-e|--extended]}} {{[-c|--offline]}}
```

# lspci

Отобразить список всех подключенных PCI-устройств.

Больше информации: https://manned.org/lspci.

- Отобразить список всех подключенных PCI-устройств:

```
lspci
```

- Показать дополнительную информацию:

```
lspci -v
```

- Отобразить драйверы и модули, работающие с каждым устройством:

```
lspci -k
```

- Отобразить определённое устройство:

```
lspci -s {{00:18.3}}
```

- Вывести информацию в удобном формате для чтения:

```
lspci -vm
```

# more

Интерактивно показать файл, позволяя прокручивать и искать текст.

Смотрите также: **`less`**.

Больше информации: https://manned.org/more.

- Открыть файл:

  `more {{путь/к/файлу}}`

- Показать, начиная с определенной строки:

  `more +{{номер_строки}} {{путь/к/файлу}}`

- Перейти на следующую страницу:

  `<Space>`

- Искать строку (нажмите `<n>`, чтобы перейти к следующему совпадению):

  `</>{{что-то}}<Enter>`

- Выйти:

  `<q>`

- Показать справку по интерактивным командам:

  `<h>`

# ncal

Эта команда — псевдоним для **`cal`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr cal
  ```

# openvpn3

Клиент OpenVPN 3 для Linux.

Больше информации: https://community.openvpn.net/openvpn/wiki/OpenVPN3Linux.

- Запустить новую VPN-сессию:

```
openvpn3 session-start {{[-c|--config]}} {{путь/к/конфигурации.conf}}
```

- Показать список установленных сессий:

```
openvpn3 sessions-list
```

- Отключить текущую сессию, запущенную с указанной конфигурацией:

```
openvpn3 session-manage {{[-c|--config]}} {{путь/к/конфигурации.conf}} {{[-D|--disconnect]}}
```

- Импортировать конфигурацию VPN:

```
openvpn3 config-import {{[-c|--config]}} {{путь/к/конфигурации.conf}}
```

- Показать список импортированных конфигураций:

```
openvpn3 configs-list
```

# pacman

Утилита для управления пакетами в Arch Linux.

Смотрите также: **`pacman-sync`**, **`pacman-remove`**, **`pacman-query`**, **`pacman-upgrade`**, **`pacman-files`**, **`pacman-database`**, **`pacman-deptest`**, **`pacman-key`**, **`pacman-mirrors`**.

Для эквивалентных команд в других менеджерах пакетов смотрите https://wiki.archlinux.org/title/Pacman/Rosetta.

Больше информации: https://manned.org/pacman.8.

- [S]инхронизировать и обновить все пакеты:

  ```
  sudo pacman -Syu
  ```

- Установить новый пакет:

  ```
  sudo pacman -S {{пакет}}
  ```

- Удалить [R] пакет и его зависимости:

  ```
  sudo pacman -Rs {{пакет}}
  ```

- И[s]кать в базе данных пакетов по `regex` или ключевому слову:

  ```
  pacman -Ss "{{шаблон_поиска}}"
  ```

- Искать в базе данных пакеты, содержащие определенный [F]айл:

  ```
  pacman -F "{{имя_файла}}"
  ```

- Показать только явно [e] установленные пакеты и их версии:

  ```
  pacman -Qe
  ```

- Показать пакеты-сироты (установленные как зависимости [d], но не требующиеся ни одному пакету):

  ```
  pacman -Qtdq
  ```

- Полностью очистить кэш `pacman`:

  ```
  sudo pacman -Scc
  ```

# phar

Создать, обновить или извлечь PHP-архивы (PHAR).

Больше информации: https://manned.org/phar.

- Добавить один или несколько файлов или каталогов в Phar-файл:

```
phar add -f {{путь/к/phar_файлу}} {{путь/к/
файлу_или_каталогу1 путь/к/файлу_или_каталогу2 ...}}
```

- Показать содержимое Phar-файла:

```
phar list -f {{путь/к/phar_файлу}}
```

- Удалить указанный файл или каталог из Phar-файла:

```
phar delete -f {{путь/к/phar_файлу}} -e {{файл_или_каталог}}
```

- Сжать или распаковать файлы и каталоги в Phar-файле:

```
phar compress -f {{путь/к/phar_файлу}} -c {{алгоритм}}
```

- Получить информацию о Phar-файле:

```
phar info -f {{путь/к/phar_файлу}}
```

- Подписать Phar-файл с использованием указанного алгоритма хеширования:

```
phar sign -f {{путь/к/phar_файлу}} -h {{алгоритм}}
```

- Подписать Phar-файл с использованием приватного ключа OpenSSL:

```
phar sign -f {{путь/к/phar_файлу}} -h openssl -y {{путь/к/
приватному_ключу}}
```

- Показать справку и доступные алгоритмы хеширования/сжатия:

```
phar help
```

# pihole

Управлять DNS-сервером для блокировки рекламы Pi-hole.

Больше информации: https://docs.pi-hole.net/main/pihole-command.

- Проверить статус Pi-hole:

```
pihole status
```

- Обновить Pi-hole и Gravity:

```
sudo pihole {{[-up|updatePihole]}}
```

- Запустить или остановить демона:

```
pihole {{enable|disable}}
```

- Обновить списки и очистить кэш без перезапуска DNS-сервера:

```
pihole reloaddns
```

- Обновить список доменов, раздающих рекламу:

```
pihole {{[-g|updateGravity]}}
```

- Разрешить или запретить указанный домен:

```
pihole {{allow|deny}} {{example.com}}
```

- Искать домен в списках:

```
pihole {{[-q|query]}} {{example.com}}
```

- Открыть лог подключений в реальном времени:

```
pihole {{[-t|tail]}}
```

# pipewire

Запустить демона PipeWire.

Больше информации: https://docs.pipewire.org/page_man_pipewire_1.html.

- Запустить демона PipeWire:

  ```
  pipewire
  ```

- Использовать другой файл конфигурации:

  ```
  pipewire --config {{путь/к/файлу.conf}}
  ```

- Установить уровень подробности (error, warn, info, debug или trace):

  ```
  pipewire -{{v|vv|...|vvvvv}}
  ```

- Показать справку:

  ```
  pipewire --help
  ```

# postfix

Программа управления почтовым агентом (MTA) Postfix.

Смотрите также: **`dovecot`**, агент доставки почты (MDA), который интегрируется с Postfix.

Больше информации: https://www.postfix.org/postfix.1.html.

- Проверить конфигурацию:

  ```
  sudo postfix check
  ```

- Проверить статус демона Postfix:

  ```
  sudo postfix status
  ```

- Запустить Postfix:

  ```
  sudo postfix start
  ```

- Корректно остановить Postfix:

  ```
  sudo postfix stop
  ```

- Очистить почтовую очередь:

  ```
  sudo postfix flush
  ```

- Перезагрузить файлы конфигурации:

  ```
  sudo postfix reload
  ```

# ss

Утилита для анализа сокетов.

Больше информации: https://manned.org/ss.8.

- Показать все TCP/UDP/RAW/UNIX сокеты:

```
ss {{[-a|--all]}} {{-t|-u|-w|-x}}
```

- Фильтровать TCP-сокеты по состояниям (показывать только указанные или исключать их):

```
ss {{state|exclude}} {{bucket|big|connected|
synchronized|...}}
```

- Показать все TCP-сокеты, подключенные к локальному HTTPS-порту (443):

```
ss {{[-t|--tcp]}} src :{{443}}
```

- Показать все TCP-сокеты, прослушивающие локальный порт 8080:

```
ss {{[-lt|--listening --tcp]}} src :{{8080}}
```

- Показать все TCP-сокеты вместе с процессами, подключенными к удаленному SSH-порту:

```
ss {{[-pt|--processes --tcp]}} dst :{{ssh}}
```

- Показать все UDP-сокеты, подключенные на определенных портах источника и назначения:

```
ss {{[-u|--udp]}} 'sport == :{{порт_источника}} and dport
== :{{порт_назначения}}'
```

- Показать все TCP IPv4-сокеты, локально подключенные в подсети 192.168.0.0/16:

```
ss {{[-4t|--ipv4 --tcp]}} src {{192.168/16}}
```

- Завершить IPv4 или IPv6 сокет-соединение с указанным IP-адресом и портом назначения:

```
ss {{[-K|--kill]}} dst {{ip_адрес}} dport = {{порт}}
```

# xbps-install

XBPS утилита по (пере)установке и обновлению пакетов.

Смотрите также: **xbps**.

Больше информации: https://manned.org/xbps-install.1.

- Установить новый пакет:

```
xbps-install {{пакет}}
```

- Синхронизировать и обновить все пакеты:

```
xbps-install {{[-S|--sync]}} {{[-u|--update]}}
```

# Osx

# cut

Вырезать поля из стандартного ввода или файлов.

Больше информации: https://keith.github.io/xcode-man-pages/cut.1.html.

- Вывести указанный диапазон символов/полей каждой строки (`-c|f 1|1,10|1-10|1-|-10` далее обозначается как `диапазон`):

```
{{команда}} | cut -{{c|f}} {{1|1,10|1-10|1-|-10}}
```

- Вывести диапазон полей каждой строки с указанным разделителем:

```
{{команда}} | cut -d "{{,}}" -f {{диапазон}}
```

- Вывести диапазон символов каждой строки указанного файла:

```
cut -c {{1}} {{path/to/file}}
```

# mac-cleanup

Современная утилита для очистки macOS от кэша и мусора.

Больше информации: https://github.com/mac-cleanup/mac-cleanup-py.

- Запуск процесса очистки:

  ```
  mac-cleanup
  ```

- Открытие экрана конфигурации утилиты:

  ```
  mac-cleanup {{[-c|--configure]}}
  ```

- Выполнение пробного запуска с показом, что будет удалено в процессе очистки, без фактического удаления:

  ```
  mac-cleanup {{[-n|--dry-run]}}
  ```

- Указать каталог с пользовательскими модулями:

  ```
  mac-cleanup {{[-p|--custom-path]}} {{путь/к/каталогу}}
  ```

- Автоматическое подтверждение всех предупреждений и принудительное продолжение работы:

  ```
  mac-cleanup {{[-f|--force]}}
  ```

Sunos

# devfsadm

Команда администрирования для **`/dev`**. Поддерживает пространство имен **`/dev`**.

Больше информации: https://www.unix.com/man-page/sunos/1m/devfsadm.

- Сканировать для новых дисков:

  ```
  devfsadm -c disk
  ```

- Очистить все оборванные ссылки `/dev` и выполнить поиск нового устройства:

  ```
  devfsadm -C -v
  ```

- Пробный-запуск - вывод того, что бы изменилось, но без произведения модификаций:

  ```
  devfsadm -C -v -n
  ```

# Windows

# cd

Отображение текущего или перемещение в другой каталог.

Больше информации: https://learn.microsoft.com/windows-server/administration/windows-commands/cd.

- Отобразить путь текущего каталога:

```
cd
```

- Перейти вверх в родительский каталог:

```
cd ..
```

- Перейти в указанный каталог на текущем диске:

```
cd {{путь\до\каталога}}
```

- Перейти в указанный каталог на другом диске ([d]rive):

```
cd /d {{C}}:{{путь\до\каталога}}
```

# choco

Менеджер пакетов Chocolatey.

Некоторые подкоманды, такие как **`install`**, **`upgrade`**, **`pin`**, имеют собственную документацию.

Больше информации: https://docs.chocolatey.org/en-us/choco/commands/.

- Установить пакет:

  ```
  choco install {{имя_пакета}}
  ```

- Обновить конкретный установленный пакет:

  ```
  choco upgrade {{имя_пакета}}
  ```

- Обновить все устаревшие пакеты с автоматическим подтверждением всех запросов:

  ```
  choco upgrade all {{[-y|--yes]}}
  ```

- Удалить пакет с автоматическим подтверждением всех запросов:

  ```
  choco uninstall {{имя_пакета}} {{[-y|--yes]}}
  ```

- Искать пакеты по имени или ключевому слову:

  ```
  choco search {{запрос}}
  ```

- Показать список всех пакетов, установленных на компьютере:

  ```
  choco list
  ```

- Показать пакеты, для которых доступны новые версии:

  ```
  choco outdated
  ```

- Установить пакет из определенного источника:

  ```
  choco install {{имя_пакета}} {{[-s|--source]}} {{источник}}
  ```

# chromium

Веб-браузер с открытым исходным кодом, в основном разрабатываемый и поддерживаемый Google.

Примечание: Возможно, вам потребуется заменить команду **`chromium`** на желаемый веб-браузер, такой как **`brave`**, **`google-chrome`**, **`microsoft-edge`**/**`msedge`**, **`opera`** или **`vivaldi`**.

Больше информации: https://www.chromium.org/developers/how-tos/run-chromium-with-flags/.

- Открыть указанный URL-адрес или файл:

```
chromium {{https://example.com|путь\к\файлу.html}}
```

- Открыть в режиме инкогнито (используйте `--inprivate` для Microsoft Edge):

```
{{chromium --incognito|msedge --inprivate}} {{example.com}}
```

- Открыть в новом окне:

```
chromium --new-window {{example.com}}
```

- Открыть в режиме приложения (без панелей инструментов, адресной строки, кнопок и т.д.):

```
chromium --app {{https://example.com}}
```

- Использовать прокси-сервер:

```
chromium --proxy-server "{{socks5://hostname:
66}}" {{example.com}}
```

- Открыть с пользовательским каталогом профиля:

```
chromium --user-data-dir {{путь\к\каталогу}}
```

- Открыть без проверки CORS (полезно для тестирования API):

```
chromium --user-data-dir {{путь\к\каталогу}} --disable-web-
security
```

- Открыть с окном DevTools для каждой новой вкладки:

```
chromium --auto-open-devtools-for-tabs
```

# cinst

Эта команда — псевдоним для **`choco install`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr choco install
  ```

# clist

Эта команда — псевдоним для **`choco list`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr choco list
  ```

# cuninst

Эта команда — псевдоним для **`choco uninstall`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr choco uninstall
  ```

# date

Просмотр и изменение текущей системной даты.

Больше информации: https://learn.microsoft.com/windows-server/administration/windows-commands/date.

- Отображение текущей системной даты, за которой следует запрос на ввод новой даты (Оставить пустым если не нужно вносить изменения):

  ```
  date
  ```

- Отображает текущую дату без запроса на новую дату:

  ```
  date /t
  ```

- Для изменения текущей даты, введите новую дату, на основе текущей конфигурации даты, а затем нажмите клавишу ВВОД:

  ```
  date {{месяц}}-{{день}}-{{год}}
  ```

# find

Поиск заданной строки в одном или нескольких файлах.

Больше информации: https://learn.microsoft.com/windows-server/administration/windows-commands/find.

- Найти строки, содержащие указанную строку:

```
find "{{строка}}" {{путь/до/файла_или_папки}}
```

- Отобразить строки, не содержащие указанную строку:

```
find "{{строка}}" {{путь/до/файла_или_папки}} /v
```

- Отобразить количество строк, содержащих указанную строку:

```
find "{{строка}}" {{путь/до/файла_или_папки}} /c
```

- Вывод номеров найденных строк:

```
find "{{строка}}" {{путь/до/файла_или_папки}} /n
```

# ftp

Интерактивно передавать файлы между локальным и удаленным FTP-сервером.

Больше информации: https://learn.microsoft.com/windows-server/administration/windows-commands/ftp.

- Интерактивно подключиться к удаленному FTP-серверу:

```
ftp {{хост}}
```

- Войти как анонимный пользователь:

```
ftp -A {{хост}}
```

- Отключить автоматический вход при первоначальном подключении:

```
ftp -n {{хост}}
```

- Выполнить файл, содержащий список FTP-команд:

```
ftp -s:{{путь\к\файлу}} {{хост}}
```

- Скачать несколько файлов (используя glob-выражение):

```
mget {{*.png}}
```

- Загрузить несколько файлов (используя glob-выражение):

```
mput {{*.zip}}
```

- Удалить несколько файлов на удаленном сервере:

```
mdelete {{*.txt}}
```

- Показать справку:

```
ftp --help
```

# ipconfig

Отображение и управление сетевыми настройками Windows.

Больше информации: https://learn.microsoft.com/windows-server/administration/windows-commands/ipconfig.

- Показать список сетевых адаптеров:

  ```
  ipconfig
  ```

- Показать подробный список сетевых адаптеров:

  ```
  ipconfig /all
  ```

- Обновить IP-адреса сетевого адаптера:

  ```
  ipconfig /renew {{адаптер}}
  ```

- Освободить IP-адреса сетевого адаптера:

  ```
  ipconfig /release {{адаптер}}
  ```

- Удалить все данные из кеша DNS:

  ```
  ipconfig /flushdns
  ```

# iwr

Эта команда — псевдоним для **`invoke-webrequest`**.

- Смотри документацию для оригинальной команды:

```
tldr invoke-webrequest
```

# mkdir

Создает каталог или подкаталог в файловой системе.

Больше информации: https://learn.microsoft.com/windows-server/administration/windows-commands/mkdir.

- Создать новый каталог:

```
mkdir {{путь\до\папки}}
```

- Чтобы создать дерево каталогов в корневом каталоге введите:

```
mkdir {{путь\до\под_папки}}
```

# msiexec

Установка, обновление, восстановление или удаление программ Windows через пакеты установки MSI и MSP.

Больше информации: https://learn.microsoft.com/windows-server/administration/windows-commands/msiexec.

- Установить программу из MSI-пакета:

  ```
  msiexec /package {{путь/до/файла.msi}}
  ```

- Установить MSI-пакет с веб-сайта:

  ```
  msiexec /package {{https://example.com/installer.msi}}
  ```

- Установить MSP-пакет с обновлением (патчем):

  ```
  msiexec /update {{путь/до/файла.msp}}
  ```

- Удалить программу или обновление, используя соответствующий пакет MSI или MSP:

  ```
  msiexec /uninstall {{путь/до/файла}}
  ```

# nvm

Установка, удаление и переключение между версиями Node.js.

Поддерживает номера версий вроде "12.8" or "v16.13.1", метки вроде "stable", "system", и т.д.

Больше информации: https://github.com/coreybutler/nvm-windows.

- Установить заданную версию Node.js:

  ```
  nvm install {{версия_node}}
  ```

- Задать версию Node.js по умолчанию (требуется запускать из-под Администратора):

  ```
  nvm use {{версия_node}}
  ```

- Вывести список всех доступных версий Node.js и подсветить версию по умолчанию:

  ```
  nvm list
  ```

- Удалить указанную версию Node.js:

  ```
  nvm uninstall {{версия_node}}
  ```

# pabcnetcclear

Препроцессор и компилятор для исходных файлов PascalABC.NET.

Больше информации: https://pascalabc.net.

- Скомпилировать файл с исходным кодом в исполняемый файл с тем же именем:

  ```
  pabcnetcclear {{путь/до/исходного_файла.pas}}
  ```

- Скомпилировать файл с исходным кодом в исполняемый файл с заданным именем:

  ```
  pabcnetcclear /Output:{{путь/до/файла.exe}} {{путь/до/
  исходного_файла.pas}}
  ```

- Скомпилировать файл с исходным кодом в исполняемый файл с тем же именем с/без отладочной информации:

  ```
  pabcnetcclear /Debug:{{0|1}} {{путь/до/исходного_файла.pas}}
  ```

- Разрешить искать модули по указанному пути при компиляции файла с исходным кодом в исполняемый файл с тем же именем:

  ```
  pabcnetcclear /SearchDir:{{путь/до/папки}} {{путь/до/
  исходного_файла.pas}}
  ```

- Скомпилировать файл с исходным кодом в исполняемый файл, определив символ условной компиляции:

  ```
  pabcnetcclear /Define:{{символ}} {{путь/до/
  исходного_файла.pas}}
  ```

# print

Вывод на печать текстового файла.

Больше информации: https://learn.microsoft.com/windows-server/administration/windows-commands/print.

- Печать текстового файла используя локальный принтер:

  ```
  print {{путь\до\папки}}
  ```

- Печать текстового файла используя сетевой принтер:

  ```
  print /d:{{принтер}} {{путь\до\папки}}
  ```

# pwsh where

Эта команда — псевдоним для **`Where-Object`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr Where-Object
  ```

# set

Отобразить или задать значение переменным окружения для текущего экземпляра CMD.

Больше информации: https://learn.microsoft.com/windows-server/administration/windows-commands/set.

- Вывести список текущих переменных окружения:

  ```
  set
  ```

- Задать переменной окружения определённое значение:

  ```
  set {{имя}}={{значение}}
  ```

- Вывести список переменных окружения, имена которых начинаются с заданной строки:

  ```
  set {{имя}}
  ```

- Запросить у пользователя значение для указанной переменной:

  ```
  set /p {{имя}}={{строка_подсказки}}
  ```

# sls

Эта команда — псевдоним для **`Select-String`**.

- Смотри документацию для оригинальной команды:

  ```
  tldr select-string
  ```

# time

Просмотр и измененение системного времени.

Больше информации: https://learn.microsoft.com/windows-server/administration/windows-commands/time.

- Отображение текущего системного времени, за которым следует запрос на ввод нового времени (Оставить пустым если не нужно вносить изменения):

  ```
  time
  ```

- Отображает текущее системное время без запроса на новое время:

  ```
  time /t
  ```

# where

Показ расположения файлов, удовлетворяющих шаблону поиска.

По умолчанию поиск производится в текущей папке и по путям в переменной окружения PATH.

Больше информации: https://learn.microsoft.com/windows-server/administration/windows-commands/where.

- Отобразить расположение файлов, соответствующих шаблону:

```
where {{шаблон_файла}}
```

- Отобразить расположение файлов, соответствующих шаблону, вместе с размером и датой:

```
where /T {{шаблон_файла}}
```

- Рекурсивно искать файлы, соответствующие шаблону, по указанному пути:

```
where /R {{path/to/directory}} {{шаблон_файла}}
```

- Только вернуть код возврата для результата поиска файла по шаблону:

```
where /Q {{шаблон_файла}}
```

# wsl

Управлять Подсистемой Windows для Linux.

Больше информации: https://learn.microsoft.com/windows/wsl/reference.

- Запустить оболочку Linux (в дистрибутиве по умолчанию):

```
wsl {{команда_оболочки}}
```

- Запустить команду Linux без использования оболочки:

```
wsl {{[-e|--exec]}} {{команда}} {{аргументы_команды}}
```

- Указать конкретный дистрибутив:

```
wsl {{[-d|--distribution]}} {{дистрибутив}}
{{команда_оболочки}}
```

- Показать список доступных дистрибутивов:

```
wsl {{[-l|--list]}}
```

- Экспортировать дистрибутив в `.tar`-файл:

```
wsl --export {{дистрибутив}}
{{путь\к\файлу_дистрибутива.tar}}
```

- Импортировать дистрибутив из `.tar`-файла:

```
wsl --import {{дистрибутив}} {{путь\к\месту_установки}}
{{путь/к/файлу_дистрибутива.tar}}
```

- Изменить версию WSL, используемую для указанного дистрибутива:

```
wsl --set-version {{дистрибутив}} {{версия}}
```

- Завершить работу Подсистемы Windows для Linux:

```
wsl --shutdown
```