tldr pages book

Simplified and community-driven man pages

Generated on Sun Oct 13 08:20:15 2024

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

Создать заархивированный 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/bugreportz.

- Создать полный заархивированный отчет об ошибках на устройстве Android:

  ```
  bugreportz
  ```

- Показать ход выполнения операции `bugreportz`:

  ```
  bugreportz -p
  ```

- Показать версию `bugreportz`:

  ```
  bugreportz -v
  ```

- Показать справку:

  ```
  bugreportz -h
  ```

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

Дамп лог (журнал) системных сообщений, включая трассировку стека при возникновении ошибки и информационные сообщения, регистрируемые приложениями.

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

- Показать системные логи:

```
logcat
```

- Записать системные логи в файл:

```
logcat -f {{путь/до/файла}}
```

- Показать строки, соответствующие регулярному выражению:

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

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://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://adbinstaller.com/commands/adb-shell-wm-5b672b17e7958178a2955538.

- Показать физический размер экрана Android-устройства:

  ```
  wm size
  ```

- Показать физическую плотность экрана Android-устройства:

  ```
  wm density
  ```

Common

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}}
```

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://www.brain-dump.org/projects/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, оптимизированная 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 --ignore-case "{{шаблон_поиска}}"
```

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

```
ack -o "{{шаблон_поиска}}"
```

- Ограничить поиск только файлами определённого типа:

```
ack --type {{ruby}} "{{шаблон_поиска}}"
```

- Не искать в файлах определённого типа:

```
ack --type no{{ruby}} "{{шаблон_поиска}}"
```

- Подсчитать общее количество найденных совпадений:

```
ack --count --no-filename "{{шаблон_поиска}}"
```

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

```
ack --count --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://github.com/nektos/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`**, но имеет цель быть быстрее.

Больше информации: https://github.com/ggreer/the_silver_searcher.

- Найти файлы, содержащие "foo", и вывести подходящие строки в контексте:

```
ag {{foo}}
```

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

```
ag {{foo}} {{путь/до/папки}}
```

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

```
ag -l {{foo}}
```

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

```
ag -i -o {{FOO}}
```

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

```
ag {{foo}} -G {{bar}}
```

- Найти файлы, содержимое которых совпадает с регулярным выражением:

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

- Найти файлы, у которых имя совпадает с "foo":

```
ag -g {{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`**.

Больше информации: https://tldp.org/LDP/abs/html/aliases.html.

- Вывести список всех псевдонимов:

```
alias
```

- Создать типовой псевдоним:

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

- Вывести команду сопоставленную с данным псевдонимом:

```
alias {{псевдоним}}
```

- Удалить псевдоним:

```
unalias {{псевдоним}}
```

- Превратить `rm` в интерактивную команду:

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

- Превратить `la` в ссылку на `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 (формат на выходе по умолчанию):

  ```
  asciidoctor {{путь/до/файла.adoc}}
  ```

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

  ```
  asciidoctor -a stylesheet {{путь/до/таблицы-стилей.css}} {{путь/до/файла.adoc}}
  ```

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

  ```
  asciidoctor --embedded {{путь/до/файла.adoc}}
  ```

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

  ```
  asciidoctor --backend {{pdf}} --require {{asciidoctor-pdf}} {{путь/до/файла.adoc}}
  ```

aspell

Интерактивная проверка орфографии.

Больше информации: http://aspell.net/.

- Проверить орфографию в одном файле:

```
aspell check {{путь/до/файла}}
```

- Вывести список неверно написанных слов из стандартного ввода:

```
cat {{файл}} | 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 {{файл}} | aspell --
personal={{персональный_список_слов.pws}} list
```

bg

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

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

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

  ```
  bg
  ```

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

  ```
  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:

```
cat {{файл}}
```

- Объединяет несколька файлов в один:

```
cat {{файл1 файл2 ...}} > {{итоговый_файл}}
```

- Добавляет несколько файлов в конец файла:

```
cat {{файл1 файл2 ...}} >> {{итоговый_файл}}
```

chmod

Изменить права доступа файлу или папке.

Больше информации: https://www.gnu.org/software/coreutils/chmod.

- 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 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 a+rX {{папка}}
```

ClamAV

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

Больше информации: https://www.clamav.net.

- Смотри документацию для оригинальной команды:

  ```
  tldr clamdscan
  ```

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/cut.

- 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
```

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 текущей директории:

```
dotnet build
```

- Скомпилировать проект или решение .NET в режиме debug:

```
dotnet build {{путь/до/проекта_или_решения}}
```

- Скомпилировать в режиме release:

```
dotnet build --configuration {{Release}}
```

- Скомпилировать без восстановления зависимостей:

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

- Скомпилировать с заданным уровнем детализации выводимой информации:

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

- Скомпилировать для заданной среды исполнения:

```
dotnet build --runtime {{идентификатор_среды_исполения}}
```

- Указать целевую папку:

```
dotnet build --output {{путь/до/папки}}
```

dotnet publish

Публикует .NET-приложение и его зависимости в папку для развёртываения на целевой системе.

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

- Скомпилировать проект .NET в режиме release:

```
dotnet publish --configuration Release {{путь/до/файла_проекта}}
```

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

```
dotnet publish --self-contained true --runtime {{идентификатор_среды_исполения}} {{путь/до/файла_проекта}}
```

- Упаковать приложение в один исполняемый файл для заданной платформы:

```
dotnet publish --runtime {{идентификатор_среды_исполения}} -p:PublishSingleFile=true {{путь/до/файла_проекта}}
```

- Обрезать неиспользуемые библиотеки чтобы уменьшить размер развёртывания приложения:

```
dotnet publish --self-contained true --runtime {{идентификатор_среды_исполения}} -p:PublishTrimmed=true {{путь/до/файла_проекта}}
```

- Скомпилировать проект .NET без восстановления зависимостей:

```
dotnet publish --no-restore {{путь/до/файла_проекта}}
```

- Указать целевую папку:

```
dotnet publish --output {{путь/до/папки}} {{путь/до/файла_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 текущей директории:

  ```
  dotnet restore
  ```

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

  ```
  dotnet restore {{путь/до/проекта_или_решения}}
  ```

- Восстановить зависимости без кеширования HTTP-запросов:

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

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

  ```
  dotnet restore --force
  ```

- Восстановить зависимости, считая что ошибки источника пакетов это предупреждения:

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

- Восстановить зависимости, используя заданный уровень детализации выводимой информации:

  ```
  dotnet restore --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

Отобразить заданные аргументы.

Больше информации: https://www.gnu.org/software/coreutils/echo.

- Отобразить текстовое сообщение. Примечание: кавычки необязательны:

```
echo "{{Привет, мир}}"
```

- Отобразить сообщение с переменной окружения:

```
echo "{{Мой путь - $PATH}}"
```

- Отобразить сообщение, не перенося каретку на новую строку:

```
echo -n "{{Привет, мир}}"
```

- Добавить сообщение в файл:

```
echo "{{Привет, мир}}" >> {{путь/к/файлу.txt}}
```

- Экранировать с помощью символа обратной косой черты (специальный символ):

```
echo -e "{{Column 1\tColumn 2}}"
```

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 --prompt='> '
```

- Запустить интерактивную сессию редактора с удобными для пользователя ошибками:

```
ed --verbose
```

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

```
ed --quiet
```

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

```
ed --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
```

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.

Больше информации: https://manned.org/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`**.

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

- Смотри документацию для оригинальной команды:

  ```
  tldr fossil-commit
  ```

fossil delete

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

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

- Смотри документацию для оригинальной команды:

  ```
  tldr fossil rm
  ```

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`**.

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

- Смотри документацию для оригинальной команды:

  ```
  tldr fossil-init
  ```

gh cs

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

Больше информации: https://cli.github.com/manual/gh_codespace.

- Смотри документацию для оригинальной команды:

```
tldr gh-codespace
```

ghc

Компилятор Glasgow Haskell Compiler.

Компиляция и компоновка исходных файлов Haskell.

Больше информации: https://www.haskell.org/ghc.

- 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`**.

Больше информации: https://gnmic.kmrd.dev/cmd/subscribe.

- Смотри документацию для оригинальной команды:

```
tldr gnmic subscribe
```

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 {{-r|--recursive}} {{-n|--line-number}} --binary-files={{without-match}} "{{шаблон_поиска}}" {{путь/к/директории}}
```

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

```
grep {{-E|--extended-regexp}} {{-i|--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 {{-H|--with-filename}} {{-n|--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}}"
```

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}}
```

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 {{путь/до/файла}}
```

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

```
hexdump -C -n{{количество_байтов}} {{путь/до/файла}}
```

- Не заменять повторяющиеся строки на '*':

```
hexdump --no-squeezing {{путь/до/файла}}
```

history expansion

Повторное использование и подстановка команд из списка истории в **sh**, Bash, Zsh, **rbash** and **ksh**.

Больше информации: https://www.gnu.org/software/bash/manual/html_node/History-Interaction.

- Запустить предыдущую команду от имени суперпользователя (`!!` заменяется на предыдущую команду):

  ```
  sudo !!
  ```

- Запустить команду с последним аргументом из предыдущей команды:

  ```
  {{команда}} !$
  ```

- Запустить команду с первым аргументом из предыдущей команды:

  ```
  {{команда}} !^
  ```

- Запустить `n`-ую с начала команду из истории:

  ```
  !{{n}}
  ```

- Запустить `n`-ую с конца команду из истории :

  ```
  !-{{n}}
  ```

- Запустить самую последнюю команду, содержащую `строка`:

  ```
  !?{{строка}}?
  ```

- Запустить предыдущую команду, заменив `строка1` на `строка2`:

  ```
  ^{{строка1}}^{{строка2}}^
  ```

- Выполнить подстановку команд из списка истории и вывести на экран получившуюся команду, не запуская её:

  ```
  {{!-n}}:p
  ```

history

История командной строки.

Больше информации: https://www.gnu.org/software/bash/manual/html_node/Bash-History-Builtins.html.

- Отобразить список истории команд с номерами строк:

  ```
  history
  ```

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

  ```
  history {{20}}
  ```

- Очистить список истории команд (только для текущей оболочки Bash):

  ```
  history -c
  ```

- Перезаписать файл истории историей текущей оболочки Bash (часто комбинируется с `history -c` для очистки истории):

  ```
  history -w
  ```

- Удалить элемент истории с указанным номером:

  ```
  history -d {{номер}}
  ```

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 командной строки, использующий доменный язык.

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

- Выполнить указанное выражение (вывести цветной и отформатированный JSON):

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

- Выполнить указанный скрипт:

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

- Передать указанные агрументы:

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

- Вывести указанные ключи:

```
{{cat путь/к/файлу.json}} | jq '{{.ключ1, .ключ2, ...}}'
```

- Вывести указанные элементы массива:

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

- Вывести все элементы массива/ключи объекта:

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

- 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/ls.

- Список файлов по одному в строке:

```
ls -1
```

- Список всех файлов, включая скрытые:

```
ls -a
```

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

```
ls -F
```

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

```
ls -la
```

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

```
ls -lh
```

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

```
ls -lS
```

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

```
ls -ltr
```

- Список только директорий:

```
ls -d */
```

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
  ```

mscore

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

Больше информации: https://musescore.org/handbook/command-line-options.

- Смотри документацию для оригинальной команды:

```
tldr musescore
```

nohup

Позволяет процессу продолжать работу после закрытия терминала.

Больше информации: https://www.gnu.org/software/coreutils/nohup.

- 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}} &
```

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`**.

Больше информации: https://docs.platformio.org/en/latest/core/userguide/.

- Смотри документацию для оригинальной команды:

```
tldr pio
```

pwd

Отобразить абсолютной путь до текущей/рабочей директории.

Больше информации: https://www.gnu.org/software/coreutils/pwd.

- Отобразить абсолютной путь до текущей директории:

```
pwd
```

- Отобразить абсолютной путь до текущей директории и "разрешить" символические ссылки (т.е. показать "фактический" путь):

```
pwd -P
```

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/rm.

- 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}}
```

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/до/файла}}
```

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

```
stat --terse {{путь/до/файла}}
```

- Показать информацию о файловой системе:

```
stat --file-system {{путь/до/файла}}
```

- Показать только права доступа в восьмеричном виде:

```
stat --format="%a %n" {{путь/до/файла}}
```

- Показать владельца и группу файла:

```
stat --format="%U %G" {{путь/до/файла}}
```

- Показать размер файла в байтах:

```
stat --format="%s %n" {{путь/до/файла}}
```

stty

Настройка параметров интерфейса терминального устройства.

Больше информации: https://www.gnu.org/software/coreutils/stty.

- Показать все настройки для текущего терминала:

```
stty --all
```

- Задать количество строк или столбцов:

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

- Получить фактическую скорость передачи данных устройства:

```
stty --file {{путь/до/файла_устройства}} speed
```

- Сбросить все режимы до разумных значений для текущего терминала:

```
stty sane
```

tar

Утилита архивирования.

Обычно используется в сочетании с методом сжатия, такими как gzip или bzip2.

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

- Создать архив из файлов:

```
tar cf {{целевой.tar}} {{файл1}} {{файл2}} {{файл3}}
```

- Создать gzip архив:

```
tar czf {{целевой.tar.gz}} {{файл1}} {{файл2}} {{файл3}}
```

- Создать gzip-архив из директории, используя относительные пути:

```
tar czf {{целевой.tar.gz}} -C {{путь/до/папки}} .
```

- Извлечь (сжатый) архив в указанную папку:

```
tar xf {{исходный.tar[.gz|.bz2|.xz]}} --directory={{папка}}
```

- Создать сжатый архив, используя суффикс архива для определения программы сжатия:

```
tar caf {{целевой.tar.xz}} {{файл1}} {{файл2}} {{файл3}}
```

- Вывести список содержимого tar-файла:

```
tar tvf {{исходный.tar}}
```

- Извлечь файлы, соответствующие шаблону:

```
tar xf {{исходный.tar}} --wildcards "{{*.html}}"
```

tldr

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

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

- Показывает типичное использование комманды (подсказка: то как вы попали сюда!):

```
tldr {{команда}}
```

- Показывает tldr страницу для комманды tar для Linux:

```
tldr -p {{linux}} {{tar}}
```

- Получить помощь по подкоманде Git:

```
tldr {{git-checkout}}
```

- Обновить локальные tldr страницы (если клиент поддерживает кэширование):

```
tldr -u
```

tldrl

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

Больше информации: https://github.com/tldr-pages/tldr-lint.

- Смотри документацию для оригинальной команды:

```
tldr tldr-lint
```

tlmgr arch

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

Больше информации: https://www.tug.org/texlive/tlmgr.html.

- 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 {{номер_строки}} {{номер_столбца}}
```

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

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

- Показать количество столбцов, строк или цветов:

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

- Подать звуковой сигнал терминала:

```
tput bel
```

- Сбросить все атрибуты терминала:

```
tput sgr0
```

- Включить или отключить перенос слов:

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

tty

Выводит название терминала.

Больше информации: https://www.gnu.org/software/coreutils/tty.

- Вывести имя файла, соответствующее текущему терминалу:

```
tty
```

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

Переводить HTML в PDF или PNG.

Больше информации: https://weasyprint.org/.

- Перевести HTML файл в PDF:

```
weasyprint {{путь/до/входного.html}} {{путь/до/
выходного.pdf}}
```

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

```
weasyprint {{путь/до/входного.html}} {{путь/до/
выходного.png}} --stylesheet {{путь/до/таблицы-стилей.css}}
```

- При переводе выводить дополнительную отладочную информацию:

```
weasyprint {{путь/до/входного.html}} {{путь/до/
выходного.pdf}} --verbose
```

- При выводе в PNG указать нестандартное разрешение:

```
weasyprint {{путь/до/входного.html}} {{путь/до/
выходного.png}} --resolution {{300}}
```

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

```
weasyprint {{путь/до/входного.html}} {{путь/до/
выходного.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 {{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 {{путь/до/архива.zip}} {{путь/до/файла_или_папки1
путь/до/файла_или_папки2 ...}}
```

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

```
zip -d {{путь/до/архива.zip}} {{путь/до/файла_или_папки1
путь/до/файла_или_папки2 ...}}
```

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

```
zip -r {{путь/до/архива.zip}} {{путь/до/файла_или_папки1
путь/до/файла_или_папки2 ...}} -x {{путь/до/
исключаемых_файлов_или_папок}}
```

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

```
zip -r -{{0-9}} {{путь/до/архива.zip}} {{путь/до/
файла_или_папки1 путь/до/файла_или_папки2 ...}}
```

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

```
zip -r -e {{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 -r -s {{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 {{путь/до/архива.zip}}
```

zsh

Z SHell — командный интерпретатор, совместимый с Bash.

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

Больше информации: https://www.zsh.org.

- 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

hostnamectl

Получение и установка имени хоста компьютера.

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

- 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
  ```

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}}
```

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
  ```

- Отобразить описание (обычно полное наименование) операционной системы:

  ```
  lsb_release -d
  ```

- Отобразить наименование ОС, без указания поля "Distributor ID":

  ```
  lsb_release -i -s
  ```

- Отобразить номер релиза (release number) и кодовое наименование дистрибутива без указания полей с названием:

  ```
  lsb_release -rcs
  ```

lsblk

Отобразить информацию об устройствах.

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

- Отобразить список всех накопителей в древовиднov виде:

```
lsblk
```

- Отобразить все устройства, в том числе "пустые":

```
lsblk -a
```

- Отобразить столбец SIZE в байтах, а не в удобночитаемом формате:

```
lsblk -b
```

- Вывод информации о файловой системе:

```
lsblk -f
```

- Использовать символы ASCII при отображении в формате дерева:

```
lsblk -i
```

- Вывести информацию о топологии блочного устройства:

```
lsblk -t
```

- Исключить устройства, указанные в списке основных номеров устройств, разделенных запятыми:

```
lsblk -e {{1,7}}
```

- Отобразить вывод с указанием списка определённых параметров, разделенных запятыми:

```
lsblk --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 --extended
```

- Отобразить информацию в виде таблицы для процессорных ядер, которые находятся в отключенном состоянии:

```
lscpu --extended --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
```

- Отобрзить определённое устройство:

```
lspci -s {{00:18.3}}
```

- Вывести информацию в удобном формате для чтения:

```
lspci -vm
```

man

Утилита просмотра справочных страницs.

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

- Показать справочную страницу для команды:

```
man {{команда}}
```

- Показать справочную страницу пакета макросов команды из раздела:

```
man {{1..9}} {{команда}}
```

- Отобразить краткое описание из справочной страницы, если оно есть:

```
man --whatis {{команда}}
```

- Отобразить путь поиска справочных страниц:

```
man --path
```

- Отобразить расположение справочной страницы, а не саму справочную страницу:

```
man --where {{команда}}
```

- Отобразить справочную страницу с использованием определённой локали:

```
man --locale={{локаль}} {{команда}}
```

- Найти справочную страницу, содержащую строку поиска:

```
man --apropos "{{строка_поиска}}"
```

ncal

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

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

- Смотри документацию для оригинальной команды:

  ```
  tldr cal
  ```

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 --sync --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}}
```

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 '> '
```

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

```
ed -s
```

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

```
ed {{путь/к/файлу}}
```

- Заменить строку указанной на всех строках:

```
,s/{{регулярное_выражение}}/{{замена}}/g
```

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}}
```

cinst

Эта команда — псевдоним для **`choco install`**.

Больше информации: https://docs.chocolatey.org/en-us/choco/commands/install.

- Смотри документацию для оригинальной команды:

```
tldr choco install
```

clist

Эта команда — псевдоним для **`choco list`**.

Больше информации: https://docs.chocolatey.org/en-us/choco/commands/list.

- Смотри документацию для оригинальной команды:

```
tldr choco list
```

cuninst

Эта команда — псевдоним для **`choco uninstall`**.

Больше информации: https://docs.chocolatey.org/en-us/choco/commands/uninstall.

- Смотри документацию для оригинальной команды:

```
tldr choco uninstall
```

curl

Эта команда — псевдоним для **`curl -p common`**.

Больше информации: https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/invoke-webrequest.

- Смотри документацию для оригинальной команды:

```
tldr curl -p common
```

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
```

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`**.

Больше информации: https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/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`**.

Больше информации: https://learn.microsoft.com/powershell/module/microsoft.powershell.core/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`**.

Больше информации: https://learn.microsoft.com/powershell/module/microsoft.powershell.utility/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}}
```