

tldr pages

Simplified and community driven man pages

airport

Wireless network configuration utility.

- Show current wireless status information:

```
airport -I
```

- Sniff wireless traffic on channel 1:

```
airport sniff {{1}}
```

- Scan for available wireless networks:

```
airport -s
```

- Disassociate from current airport network:

```
sudo airport -z
```

apachectl

Apache HTTP Server control interface for macOS.

- Start the org.apache.httpd launchd job:

```
apachectl start
```

- Stop the launchd job:

```
apachectl stop
```

- Stop, then start launchd job:

```
apachectl restart
```

archey

Simple tool for stylishly displaying system information.

- Show system information:

```
archey
```

- Show system information without colored output:

```
archey --nocolor
```

- Show system information, using MacPorts instead of Homebrew:

```
archey --macports
```

- Show system information without IP address check:

```
archey --offline
```

base64

Encode and decode using Base64 representation.

- Encode a file:

```
base64 -i {{plain_file}}
```

- Decode a file:

```
base64 -D -i {{base64_file}}
```

- Encode from stdin:

```
echo -n {{plain_text}} | base64
```

- Decode from stdin:

```
echo -n {{base64_text}} | base64 -D
```

brew cask

Package manager for macOS applications distributed as binaries.

- Search for casks:

```
brew cask search {{text}}
```

- Install a cask:

```
brew cask install {{cask_name}}
```

- List all installed casks:

```
brew cask list
```

- List installed casks that have newer versions available:

```
brew cask outdated
```

- Upgrade an installed cask to its latest version:

```
brew cask reinstall {{cask_name}}
```

- Uninstall a cask:

```
brew cask uninstall {{cask_name}}
```

- Display information about a given cask:

```
brew cask info {{cask_name}}
```

brew mas

Mas is a simple command line interface for the Mac App Store.

- Search the Mac App Store by app name and return matching identifiers:

```
mas search {{app_name}}
```

- Install or update a previously purchased application:

```
mas install {{app_name}} {{app_identifier}}
```

- Show all installed applications and their product identifiers:

```
mas list
```

- List installed apps with pending updates:

```
mas outdated
```

- Install all pending updates:

```
mas upgrade
```

- Install updates for a specific app:

```
mas upgrade {{app_identifier}}
```

brew

Package manager for macOS.

- Search for available formulas:

```
brew search {{text}}
```

- Install the latest stable version of a formula (use `--devel` for development versions):

```
brew install {{formula}}
```

- List all installed formulae:

```
brew list
```

- Update an installed formula (if no formula name is given, all installed formulae are updated):

```
brew upgrade {{formula}}
```

- Fetch the newest version of Homebrew and all formulae from GitHub:

```
brew update
```

- Show formulae that have a more recent version available:

```
brew outdated
```

- Display information about a formula (version, installation path, dependencies, etc.):

```
brew info {{formula}}
```

- Check your Homebrew installation for potential problems:

```
brew doctor
```


caffeinate

Prevent mac from sleeping.

- Prevent from sleeping for 1 hour (3600 seconds):

```
caffeinate -u -t {{3600}}
```

- Prevent from sleeping until a command completes:

```
caffeinate -s {{command}}
```

- Prevent from sleeping until you type Ctrl-C:

```
caffeinate -i
```

cal

Prints calendar information.

- Display a calendar for the current month:

```
cal
```

- Display a calendar for a specific month (1-12 or name):

```
cal -m {{month}}
```

- Display a calendar for the current year:

```
cal -y
```

- Display a calendar for a specific year (4 digits):

```
cal {{year}}
```

- Display a calendar for a specific month and year:

```
cal {{month}} {{year}}
```

- Display date of Easter (Western Christian churches) in a given year:

```
ncal -e {{year}}
```

carthage

A dependency management tool for Cocoa applications.

- Download the latest version of all dependencies mentioned in Cartfile, and build them:

```
carthage update
```

- Update dependencies, but only build for iOS:

```
carthage update --platform ios
```

- Update dependencies, but don't build any of them:

```
carthage update --no-build
```

- Download and rebuild the current version of dependencies (without updating them):

```
carthage bootstrap
```

- Rebuild a specific dependency:

```
carthage build {{dependency}}
```

clear

Clears the screen of the terminal.

- Clear the screen (equivalent to typing Control-L when using the bash shell):

`clear`

codesign

Create and manipulate code signatures for macOS.

- Sign an application with a certificate:

```
codesign -s {"My Company Name"} {/path/to/App.app}
```

- Verify the certificate of an application:

```
codesign -v {/path/to/App.app}
```

date

Set or display the system date.

- Display the current date using the default locale's format:

```
date +%c
```

- Display the current date in UTC and ISO 8601 format:

```
date -u +%Y-%m-%dT%H:%M:%SZ
```

- Display the current date as a Unix timestamp (seconds since the Unix epoch):

```
date +%s
```

- Display a specific date (represented as a Unix timestamp) using the default format:

```
date -r 1473305798
```

dd

Convert and copy a file.

- Make a bootable usb drive from an isohybrid file (such like archlinux-xxx.iso):

```
dd if={{file.iso}} of=/dev/{{usb_drive}}
```

- Clone a drive to another drive with 4MB block and ignore error:

```
dd if=/dev/{{source_drive}} of=/dev/{{dest_drive}} bs=4m  
conv=noerror
```

- Generate a file of 100 random bytes by using kernel random driver:

```
dd if=/dev/urandom of={{random_file}} bs=100 count=1
```

- Benchmark the write performance of a disk:

```
dd if=/dev/zero of={{file_1GB}} bs=1024 count=1000000
```

defaults

Read and write macOS user configuration for applications.

- Read system defaults for an application option:

```
defaults read {{application}} {{option}}
```

- Read default values for an application option:

```
defaults read -app {{application}} {{option}}
```

- Write the default value of an application option:

```
defaults write {{application}} {{option}} {{-type}} {{value}}
```

- Speed up Mission Control animations:

```
defaults write com.apple.Dock expose-animation-duration -  
float 0.1
```

- Delete all defaults of an application:

```
defaults delete {{application}}
```


diskutil

Utility to manage local disks and volumes.

- List all currently available disks, partitions and mounted volumes:

```
diskutil list
```

- Repair the file system data structures of a volume:

```
diskutil repairVolume {{/dev/diskX}}
```

- Unmount a volume:

```
diskutil unmountDisk {{/dev/diskX}}
```

- Eject a CD/DVD (unmount first):

```
diskutil eject {{/dev/disk1}}
```

ditto

Copy files and folders.

- Overwrite contents of destination folder with contents of source folder:

```
ditto {{path/to/source}} {{path/to/destination}}
```

- Print a line to the Terminal window for every file that's being copied:

```
ditto -V {{path/to/source}} {{path/to/destination}}
```

- Copy a given file or folder, while retaining the original file permissions:

```
ditto -rsrc {{path/to/source}} {{path/to/destination}}
```

dmesg

Write the kernel messages to standard output.

- Show kernel messages:

```
dmesg
```

- Show how much physical memory is available on this system:

```
dmesg | grep -i memory
```

- Show kernel messages 1 page at a time:

```
dmesg | less
```

drutil

Interact with DVD burners.

- Eject a disk from the drive:

```
drutil eject
```

- Burn a folder as an ISO9660 filesystem onto a DVD. Don't verify and eject when complete:

```
drutil burn -noverify -eject -iso9660
```

du

Disk usage: estimate and summarize file and folder space usage.

- List the sizes of a folder and any subfolders, in the given unit (KB/MB/GB):

```
du -{{k|m|g}} {{path/to/folder}}
```

- List the sizes of a folder and any subfolders, in human-readable form (i.e. auto-selecting the appropriate unit for each size):

```
du -h {{path/to/folder}}
```

- Show the size of a single folder, in human readable units:

```
du -sh {{path/to/folder}}
```

- List the human-readable sizes of a folder and of all the files and folders within it:

```
du -ah {{path/to/folder}}
```

- List the human-readable sizes of a folder and any subfolders, up to N levels deep:

```
du -h -d {{N}} {{path/to/folder}}
```

- List the human-readable size of all .jpg files in subfolders of the current folder, and show a cumulative total at the end:

```
du -ch */*.jpg
```

export

Command to mark shell variables in the current environment to be exported with any newly forked child processes.

- Set a new environment variable:

```
export {{VARIABLE}}={{value}}
```

- Remove an environment variable:

```
export -n {{VARIABLE}}
```

- Append something to the PATH variable:

```
export PATH=$PATH:{{path/to/append}}
```

feh

Lightweight image viewing utility.

- View images locally or using a URL:

```
feh {{path/to/images}}
```

- View images recursively:

```
feh --recursive {{path/to/images}}
```

- View images without window borders:

```
feh --borderless {{path/to/images}}
```

- Exit after the last image:

```
feh --cycle-once {{path/to/images}}
```

- Set the slideshow cycle delay:

```
feh --slideshow-delay {{seconds}} {{path/to/images}}
```

- Set your wallpaper (centered, filled, maximized, scaled or tiled):

```
feh --bg-{{center|fill|max|scale|tile}} {{path/to/image}}
```

file

Determine file type.

- Give a description of the type of the specified file. Works fine for files with no file extension:

```
file {{filename}}
```

- Look inside a zipped file and determine the file type(s) inside:

```
file -z {{foo.zip}}
```

- Allow file to work with special or device files:

```
file -s {{filename}}
```

- Don't stop at first file type match; keep going until the end of the file:

```
file -k {{filename}}
```

- Determine the mime encoding type of a file:

```
file -I {{filename}}
```


fsck

Check the integrity of a filesystem or repair it. The filesystem should be unmounted at the time the command is run.

It is a wrapper that calls **fsck_hfs**, **fsck_apfs**, **fsck_msdos**, **fsck_exfat**, and **fsck_udf** as needed.

- Check filesystem /dev/sda, reporting any damaged blocks:

```
fsck {{/dev/sda}}
```

- Check filesystem /dev/sda only if it is clean, reporting any damaged blocks and interactively letting the user choose to repair each one:

```
fsck -f {{/dev/sda}}
```

- Check filesystem /dev/sda only if it is clean, reporting any damaged blocks and automatically repairing them:

```
fsck -fy {{/dev/sda}}
```

- Check filesystem /dev/sda, reporting whether it has been cleanly unmounted:

```
fsck -q {{/dev/sda}}
```

head

Output the first part of files.

- Output the first few lines of a file:

```
head -n {{count_of_lines}} {{filename}}
```

- Output the first few bytes of a file:

```
head -c {{number_in_bytes}} {{filename}}
```

hostname

Show or set the system's host name.

- Show current host name:

```
hostname
```

- Set current host name:

```
hostname {{new_hostname}}
```

imgcat

A utility to display images directly on the command line.

Requires a compatible terminal such as iTerm2.

- Display an image on the command line:

```
imgcat {{filename}}
```

launchctl

A command-line interface to Apple's **launchd** manager for launch daemons (system-wide services) and launch agents (per-user programs).

launchd loads XML-based ***.plist** files placed in the appropriate locations, and runs the corresponding commands according to their defined schedule.

- Activate a user-specific agent to be loaded into **launchd** whenever the user logs in:

```
launchctl load ~/Library/LaunchAgents/{{my_script}}.plist
```

- Activate an agent which requires root privileges to run and/or should be loaded whenever any user logs in (note the absence of **~** in the path):

```
sudo launchctl load /Library/LaunchAgents/{{root_script}}.plist
```

- Activate a system-wide daemon to be loaded whenever the system boots up (even if no user logs in):

```
sudo launchctl load /Library/LaunchDaemons/{{system_daemon}}.plist
```

- Show all loaded agents/daemons, with the PID if the process they specify is currently running, and the exit code returned the last time they ran:

```
launchctl list
```

- Unload a currently loaded agent, e.g. to make changes (note: the plist file is automatically loaded into **launchd** after a reboot and/or logging in):

```
launchctl unload ~/Library/LaunchAgents/{{my_script}}.plist
```

- Manually run a known (loaded) agent/daemon, even if it isn't the right time (note: this command uses the agent's label, rather than the filename):

```
launchctl start {{my_script}}
```

- Manually kill the process associated with a known agent/daemon, if it's running:

```
launchctl stop {{my_script}}
```

lldb

The LLVM Low-Level Debugger.

- Debug an executable:

```
lldb {{executable}}
```

- Attach **lldb** to a running process with a given PID:

```
lldb -p {{pid}}
```

- Wait for a new process to launch with a given name, and attach to it:

```
lldb -w -n {{process_name}}
```

locate

Find filenames quickly.

- Look for pattern in the database. Note: the database is recomputed periodically (usually weekly or daily):

```
locate {{pattern}}
```

- Look for a file by its exact filename (a pattern containing no globbing characters is interpreted as `*pattern*`):

```
locate */{{filename}}
```

- Recompute the database. You need to do it if you want to find recently added files:

```
sudo /usr/libexec/locate.updatedb
```

logger

Add messages to syslog (/var/log/syslog).

- Log a message to syslog:

```
logger {{message}}
```

- Take input from stdin and log to syslog:

```
echo {{log_entry}} | logger
```

- Send the output to a remote syslog server running at a given port. Default port is 514:

```
echo {{log_entry}} | logger -h {{hostname}} -P {{port}}
```

- Use a specific tag for every line logged. Default is the name of logged in user:

```
echo {{log_entry}} | logger -t {{tag}}
```

- Log messages with a given priority. Default is `user.notice`. See `man logger` for all priority options:

```
echo {{log_entry}} | logger -p {{user.warning}}
```


look

Look for lines in sorted file.

- Look for lines which begins with the given prefix:

```
look {{prefix}} {{file}}
```

- Look for lines ignoring case:

```
look -f {{prefix}} {{file}}
```

md5

Calculate MD5 cryptographic checksums.

- Calculate the MD5 checksum for a file:

```
md5 {{filename}}
```

- Calculate MD5 checksums for multiple files:

```
md5 {{filename1}} {{filename2}}
```

- Output only the md5 checksum (no filename):

```
md5 -q {{filename}}
```

- Print a checksum of the given string:

```
md5 -s {{string}}
```

mdfind

List files matching a given query.

- Find a file by its name:

```
mdfind -name {{file}}
```

- Find a file by its content:

```
mdfind {{query}}
```

- Find a file containing a string, in a given directory:

```
mdfind -onlyin {{directory}} {{query}}
```

n

Tool to manage multiple node versions.

- Install a given version of node. If the version is already installed, it will be activated:

```
n {{version}}
```

- Display installed versions and interactively activate one of them:

```
n
```

- Remove a version:

```
n rm {{version}}
```

- Execute a file with a given version:

```
n use {{version}} {{file.js}}
```

- Output binary path for a version:

```
n bin {{version}}
```

netstat

Displays various networks related information such as open connections, open socket ports etc.

- List all ports:

```
netstat -a
```

- List all listening ports:

```
netstat -l
```

- List listening TCP ports:

```
netstat -t
```

- Display PID and program names for a specific port:

```
netstat -p {PROTOCOL}
```

- List information continuously:

```
netstat -c
```

networksetup

Configuration tool for Network System Preferences.

- List available network service providers (Ethernet, Wi-Fi, Bluetooth, etc):

```
networksetup -listallnetworkservices
```

- Show network settings for a particular networking device:

```
networksetup -getinfo {"Wi-Fi"}
```

- Get currently connected Wi-Fi network name (Wi-Fi device usually en0 or en1):

```
networksetup -getairportnetwork {{en0}}
```

- Connect to a particular Wi-Fi network:

```
networksetup -setairportnetwork {{en0}} {"Airport Network  
SSID"}} {{password}}
```

nm

List symbol names in object files.

- List global (extern) functions in a file (prefixed with T):

```
nm -g {{file.o}}
```

- List only undefined symbols in a file:

```
nm -u {{file.o}}
```

- List all symbols, even debugging symbols:

```
nm -a {{file.o}}
```

- Demangle C++ symbols:

```
nm {{file.o}} | c++filt
```

open

Opens files, directories and applications.

- Open a file with the associated application:

```
open {{file.ext}}
```

- Run a graphical macOS application:

```
open -a {{Application}}
```

- Run a graphical macOS app based on the bundle identifier (refer to [osascript](#) for an easy way get this):

```
open -b {{com.domain.application}}
```

- Open the current directory in Finder:

```
open .
```

- Reveal a file in finder:

```
open -R {{path/to/file}}
```

- Open all the files of a given extension in the current directory with the associated application:

```
open {{*.ext}}
```


opensnoop

Tool that tracks file opens on your system.

- Print all file opens as they occur:

```
sudo opensnoop
```

- Track all file opens by a process by name:

```
sudo opensnoop -n {{process_name}}
```

- Track all file opens by a process by PID:

```
sudo opensnoop -p {{PID}}
```

- Track which processes open a specified file:

```
sudo opensnoop -f {{path/to/file}}
```

osascript

Run AppleScript or JavaScript for Automation (JXA) from the command line.

- Run an AppleScript command:

```
osascript -e '{{say "Hello world"}}'
```

- Run multiple AppleScript commands:

```
osascript -e '{{say "Hello"}}' -e '{{say "world"}}'
```

- Run a compiled (*.scpt), bundled (*.scptd), or plaintext (*.applescript) AppleScript file:

```
osascript {{path/to/apple.scpt}}
```

- Get the bundle identifier of an application (useful for `open -b`):

```
osascript -e 'id of app "{{Application}}"'
```

- Run a JavaScript command:

```
osascript -l JavaScript -e '{{console.log("Hello world");}}'
```

- Run a JavaScript file:

```
osascript -l JavaScript {{path/to/script.js}}
```

pbcopy

Place standard output in the clipboard.

- Place the contents of a file in the clipboard:

```
pbcopy < {{file}}
```

- Place the results of a command in the clipboard:

```
find . -type t -name "*.png" | pbcopy
```

pbpaste

Send the contents of the clipboard to standard output.

- Write the contents of the clipboard to a file:

```
pbpaste > {{file}}
```

- Use the contents of the clipboard as input to a command:

```
pbpaste | grep foo
```

pdftoprep

Search text in PDF files.

- Find lines that match pattern in a PDF:

```
pdftoprep {{pattern}} {{file.pdf}}
```

- Include file name and page number for each matched line:

```
pdftoprep --with-filename --page-number {{pattern}}  
{{file.pdf}}
```

- Do a case insensitive search for lines that begin with "foo" and return the first 3 matches:

```
pdftoprep --max-count {{3}} --ignore-case {'^foo'}  
{{file.pdf}}
```

- Find pattern in files with a .pdf extension in the current directory recursively:

```
pdftoprep --recursive {{pattern}}
```

- Find pattern on files that match a specific glob in the current directory recursively:

```
pdftoprep --recursive --include {'*book.pdf'} {{pattern}}
```

pmset

Configure macOS power management settings, as one might do in System Preferences > Energy Saver.

Commands that modify settings must begin with **sudo**.

- Display the current power management settings:

```
pmset -g
```

- Display the current power source and battery levels:

```
pmset -g batt
```

- Set display to never sleep when on charger power:

```
sudo pmset -c displaysleep 0
```

- Set display to sleep after 15 minutes when on battery power:

```
sudo pmset -b displaysleep 15
```

- Schedule computer to automatically wake up every weekday at 9 AM:

```
sudo pmset repeat wake MTWRF 09:00:00
```

- Restore to system defaults:

```
sudo pmset -a displaysleep 10 disksleep 10 sleep 30 womp 1
```

pod

Dependency manager for Swift and Objective-C Cocoa projects.

- Create a Podfile for the current project with the default contents:

```
pod init
```

- Download and install all pods defined in the Podfile (that haven't been installed before):

```
pod install
```

- List all available pods:

```
pod list
```

- Show the outdated pods (of those currently installed):

```
pod outdated
```

- Update all currently installed pods to their newest version:

```
pod update
```

- Update a specific (previously installed) pod to its newest version:

```
pod update {{pod_name}}
```

- Remove CocoaPods from a Xcode project:

```
pod deintegrate {{xcode_project}}
```

popd

Remove a directory placed on the directory stack via the pushd shell built-in.

- Remove the top directory from the stack and cd to it:

```
popd
```

- Remove the Nth directory (starting from zero to the left from the list printed with `dirs`):

```
popd +N
```

- Remove the Nth directory (starting from zero to the right from the list printed with `dirs`):

```
popd -N
```


port

Package manager for macOS.

- Search for a package:

```
port search {{search_term}}
```

- Install a package:

```
sudo port install {{package_name}}
```

- List installed packages:

```
port installed
```

- Update port and fetch latest list of available packages:

```
sudo port selfupdate
```

- Upgrade outdated packages:

```
sudo port upgrade outdated
```

- Remove old versions of installed packages:

```
sudo port uninstall inactive
```

pushd

Place a directory on a stack so it can be accessed later.

See also **popd** to switch back to original directory.

- Switch to directory and push it on the stack:

```
pushd {{directory}}
```

- Switch first and second directories on the stack:

```
pushd
```

- Rotate stack by making the 5th element the top of the stack:

```
pushd +4
```

pwgen

Generate pronounceable passwords.

- Generate random password with s[y]mbols:

```
pwgen -y {{length}}
```

- Generate secure, hard-to-memorize passwords:

```
pwgen -s {{length}}
```

- Generate password with at least one capital letter in them:

```
pwgen -c {{length}}
```

qlmanage

QuickLook server tool.

- Display QuickLook for one or multiple files:

```
qlmanage -p {{filename}} {{filename2}}
```

- Compute 300px wide PNG thumbnails of all JPEGs in the current directory and put them in a directory:

```
qlmanage {{*.jpg}} -t -s {{300}} {{path/to/directory}}
```

- Reset Quicklook:

```
qlmanage -r
```

route

Manually manipulate the routing tables.

Necessitates to be root.

- Add a route to a destination through a gateway:

```
sudo route add {{dest_ip_address}} {{gateway_address}}
```

- Add a route to a /24 subnet through a gateway:

```
sudo route add {{subnet_ip_address}}/24 {{gateway_address}}
```

- Run in test mode (does not do anything, just print):

```
sudo route -t add {{dest_ip_address}}/24 {{gateway_address}}
```

- Remove all routes:

```
sudo route flush
```

- Delete a specific route:

```
sudo route delete {{dest_ip_address}}/24
```

- Lookup and display the route for a destination (hostname or IP address):

```
sudo route get {{destination}}
```

runit

3-stage init system.

- Start runit's 3-stage init scheme:

```
runit
```

- Shut down runit:

```
kill --CONT {{runit_pid}}
```

runsv

Start and manage a runit service.

- Start a runit service as the current user:

```
runsv {{path/to/service}}
```

- Start a runit service as root:

```
sudo runsv {{path/to/service}}
```

runsvchdir

Change the directory **runsvdir** uses by default.

- Switch **runsvdir** directories:

```
sudo runsvchdir {{/path/to/directory}}
```


runsvdir

Run an entire directory of services.

- Start and manage all services in a directory as the current user:

```
runsvdir {{path/to/services}}
```

- Start and manage all services in a directory as root:

```
sudo runsvdir {{path/to/services}}
```

- Start services in separate sessions:

```
runsvdir -P {{path/to/services}}
```

say

Converts text to speech.

- Say a phrase aloud:

```
say {"I like to ride my bike."}
```

- Read a file aloud:

```
say -f {{filename.txt}}
```

- Say a phrase with a custom voice and speech rate:

```
say -v {{voice}} -r {{words_per_minute}} {"I'm sorry Dave, I  
can't let you do that."}
```

- List the available voices:

```
say -v ?
```

- Create an audio file of the spoken text:

```
say -o {{filename.aiff}} {"Here's to the Crazy Ones."}
```

scutil

Manage system configuration parameters.

Necessitates to be root when setting configuration.

- Display DNS Configuration:

```
scutil --dns
```

- Display proxy configuration:

```
scutil --proxy
```

- Get computer name:

```
scutil --get ComputerName
```

- Set computer name:

```
sudo scutil --set ComputerName {{computer_name}}
```

- Get hostname:

```
scutil --get HostName
```

- Set hostname:

```
scutil --set HostName {{hostname}}
```

sed

Run replacements based on regular expressions.

- Replace the first occurrence of a string in a file, and print the result:

```
sed 's/{{find}}/{{replace}}/' {{filename}}
```

- Replace all occurrences of an extended regular expression in a file:

```
sed -E 's/{{regex}}/{{replace}}/g' {{filename}}
```

- Replace all occurrences of a string in a file, overwriting the file (i.e. in-place):

```
sed -i '' 's/{{find}}/{{replace}}/g' {{filename}}
```

- Replace only on lines matching the line pattern:

```
sed '/{{line_pattern}}/s/{{find}}/{{replace}}/' {{filename}}
```

- Apply multiple find-replace expressions to a file:

```
sed -e 's/{{find}}/{{replace}}/' -e 's/{{find}}/{{replace}}/'  
{{filename}}
```

- Replace separator / by any other character not used in the find or replace patterns, e.g., #:

```
sed 's#{{find}}#{{replace}}#' {{filename}}
```

shasum

Calculate or check cryptographic SHA checksums.

- Calculate the SHA1 checksum for a file:

```
shasum {{filename}}
```

- Calculate the SHA256 checksum for a file:

```
shasum --algorithm 256 {{filename}}
```

- Calculate the SHA512 checksum for multiple files:

```
shasum --algorithm 512 {{filename1}} {{filename2}}
```

- Check a file with a list of sums against the directory's files:

```
shasum --check {{list_file}}
```

- Calculate the SHA1 checksum from stdin:

```
{{somecommand}} | shasum
```

shutdown

Shutdown and reboot the system.

- Power off (halt) immediately:

```
shutdown -h now
```

- Sleep immediately:

```
shutdown -s now
```

- Reboot immediately:

```
shutdown -r now
```

- Reboot in 5 minutes:

```
shutdown -r +{{5}}
```

softwareupdate

A tool for updating MacOS App Store apps via the command line.

- List all available updates:

```
softwareupdate -l
```

- Download and install all updates:

```
softwareupdate -ia
```

- Download and install all recommended updates:

```
softwareupdate -ir
```

- Download and install a specific app:

```
softwareupdate -i {{update_name}}
```

ssh-add

Manage loaded ssh keys in the ssh-agent.

Ensure that ssh-agent is up and running for the keys to be loaded in it.

- Add the default ssh keys in "~/.ssh" to the ssh-agent:

```
ssh-add
```

- Add a specific key to the ssh-agent:

```
ssh-add {{path/to/private_key}}
```

- List fingerprints of currently loaded keys:

```
ssh-add -l
```

- Delete a key from the ssh-agent:

```
ssh-add -d {{path/to/private_key}}
```

- Delete all currently loaded keys from the ssh-agent:

```
ssh-add -D
```

- Add a key to the ssh-agent and the keychain:

```
ssh-add -K {{path/to/private_key}}
```


sshuttle

Transparent proxy server that tunnels traffic over an SSH connection.

Doesn't require admin, or any special setup on the remote SSH server.

- Forward all IPv4 TCP traffic via a remote SSH server:

```
sshuttle --remote={{username}}@{{sshserver}} {{0.0.0.0/0}}
```

- Forward all IPv4 TCP and DNS traffic:

```
sshuttle --dns --remote={{username}}@{{sshserver}}  
{{0.0.0.0/0}}
```

- Use the tproxy method to forward all IPv4 and IPv6 traffic:

```
sudo sshuttle --method=tproxy --remote={{username}}@  
{{sshserver}} {{0.0.0.0/0}} {{::/0}} --exclude=  
{{your_local_ip_address}} --exclude={{ssh_server_ip_address}}
```

stat

Display file status.

- Show file properties such as size, permissions, creation and access dates among others:

```
stat {{file}}
```

- Same as above but verbose (more similar to linux's **stat**):

```
stat -x {{file}}
```

- Show only octal file permissions:

```
stat -f %Mp%Lp {{file}}
```

- Show owner and group of the file:

```
stat -f "%Su %Sg" {{file}}
```

- Show the size of the file in bytes:

```
stat -f "%z %N" {{file}}
```

SV

Control a running runsv service.

- Start a service:

```
sudo sv up {{path/to/service}}
```

- Stop a service:

```
sudo sv down {{path/to/service}}
```

- Get service status:

```
sudo sv status {{path/to/service}}
```

sw_vers

Print macOS Software versioning information.

- Print macOS Version:

```
sw_vers -productVersion
```

- Print macOS Build:

```
sw_vers -buildVersion
```

sysctl

Access kernel state information.

- Show all available variables and their values:

```
sysctl -a
```

- Show Apple model identifier:

```
sysctl -n hw.model
```

- Show CPU model:

```
sysctl -n machdep.cpu.brand_string
```

- Show available CPU features (MMX, SSE, SSE2, SSE3, AES, etc):

```
sysctl -n machdep.cpu.feature
```

- Set a changeable kernel state variable:

```
sysctl -w {{section.tunable}}={{value}}
```

system_profiler

Report system hardware and software configuration.

- Display a full system profiler report which can be opened by System Profiler.app:

```
system_profiler -xml > MyReport.spx
```

- Display a hardware overview (Model, CPU, Memory, Serial, etc):

```
system_profiler SPHardwareDataType
```

- Print the system serial number:

```
system_profiler SPHardwareDataType | grep "Serial Number  
(system)" | awk '{print $4}'
```

systemsetup

Configure System Preferences machine settings.

- Enable remote login (SSH):

```
systemsetup -setremotelogin on
```

- Specify TimeZone, NTP Server and enable network time:

```
systemsetup -settimezone {{US/Pacific}} -setnetworktimeserver  
{{us.pool.ntp.org}} -setusingnetworktime on
```

- Make the machine never sleep and automatically restart on power failure or kernel panic:

```
systemsetup -setsleep off -setrestartpowerfailure on -  
setrestartfreeze on
```

- List valid startup disks:

```
systemsetup -liststartupdisks
```

- Specify a new startup disk:

```
systemsetup -setstartupdisk {{path}}
```

top

Display dynamic real-time information about running processes.

- Start top, all options are available in the interface:

```
top
```

- Start top sorting processes by internal memory size (default order - process ID):

```
top -o mem
```

- Start top sorting processes first by CPU, then by running time:

```
top -o cpu -0 time
```

- Start top displaying only processes owned by given user:

```
top -user {{user_name}}
```

- Get help about interactive commands:

```
?
```


tree

Show the contents of the current directory as a tree.

- Show files and directories up to 'num' levels of depth (where 1 means the current directory):

```
tree -L {{num}}
```

- Show directories only:

```
tree -d
```

- Show hidden files too:

```
tree -a
```

- Print the tree without indentation lines, showing the full path instead (use **-N** to not escape whitespace and special characters):

```
tree -i -f
```

- Print the size of each node next to it, in human-readable format, with folders displaying their cumulative size (as in the **du** command):

```
tree -s -h --du
```

- Find files within the tree hierarchy, using a wildcard (glob) pattern, and pruning out directories that don't contain matching files:

```
tree -P '{{*.txt}}' --prune
```

- Find directories within the tree hierarchy, pruning out directories that aren't ancestors of the wanted one:

```
tree -P {{directory_name}} --matchdirs --prune
```

uname

Print details about the current machine and the operating system running on it.

Note: for additional information about the operating system, try the **sw_vers** command.

- Print hardware-related information: machine and processor:

```
uname -mp
```

- Print software-related information: operating system, release number, and version:

```
uname -s rv
```

- Print the nodename (hostname) of the system:

```
uname -n
```

- Print all available system information (hardware, software, nodename):

```
uname -a
```

W

Show who is logged on and what they are doing.

Print user login, TTY, remote host, login time, idle time, current process.

- Show logged-in users info:

```
w
```

- Show logged-in users info without a header:

```
w -h
```

- Show info about logged-in users, sorted by their idle time:

```
w -i
```

wacaw

A little command-line tool for macOS that allows you to capture both still pictures and video from an attached camera.

- Take a picture from webcam:

```
wacaw {{filename}}
```

- Record a video:

```
wacaw --video {{filename}} -D {{duration_in_seconds}}
```

- Take a picture with custom resolution:

```
wacaw -x {{width}} -y {{height}} {{filename}}
```

- Copy image just taken to clipboard:

```
wacaw --to-clipboard
```

- List the devices available:

```
wacaw -L
```

xattr

Utility to work with extended filesystem attributes.

- List key:value extended attributes for a given file:

```
xattr -l {{file}}
```

- Write an attribute for a given file:

```
xattr -w {{attribute_key}} {{attribute_value}} {{file}}
```

- Delete an attribute from a given file:

```
xattr -d {{com.apple.quarantine}} {{file}}
```

- Delete all extended attributes from a given file:

```
xattr -c {{file}}
```

- Recursively delete an attribute in a given directory:

```
xattr -rd {{attribute_key}} {{directory}}
```

xcodebuild

Build Xcode projects.

- Build workspace:

```
xcodebuild -workspace {{workspace_name.workspace}} -scheme  
{{scheme_name}} -configuration {{configuration_name}} clean  
build SYMROOT={{SYMROOT_path}}
```

- Build project:

```
xcodebuild -target {{target_name}} -configuration  
{{configuration_name}} clean build SYMROOT={{SYMROOT_path}}
```

- Show SDKs:

```
xcodebuild -showsdk
```

xctool

Tool for building Xcode projects.

- Build a single project without any workspace:

```
xctool -project {{YourProject.xcodeproj}} -scheme  
{{YourScheme}} build
```

- Build a project that is part of a workspace:

```
xctool -workspace {{YourWorkspace.xcworkspace}} -scheme  
{{YourScheme}} build
```

- Clean, build and execute all the tests:

```
xctool -workspace {{YourWorkspace.xcworkspace}} -scheme  
{{YourScheme}} clean build test
```

xed

Opens files for editing in XCode.

- Open file in XCode:

```
xed {{file1}}
```

- Open file(s) in XCode, create if it doesn't exist:

```
xed -c {{filename1}}
```

- Open a file in XCode and jump to line number 75:

```
xed -l 75 {{filename}}
```


xsltproc

Transform XML with XSLT to produce output (usually HTML or XML).

- Transform an XML file with a specific XSLT stylesheet:

```
xsltproc --output {{output.html}} {{stylesheet.xslt}}  
{{xmlfile.xml}}
```

- Pass a value to a parameter in the stylesheet:

```
xsltproc --output {{output.html}} --stringparam {{name}}  
{{value}} {{stylesheet.xslt}} {{xmlfile.xml}}
```