TLDR Pages

The Book

Simplified and community-driven man pages

`tldr-pages.github.io`

Contents

1 COMMON

7za

A file archiver with high compression ratio.

- Compress directory or file:

```
7za a compressed.7z directory_or_file_to_compress
```

- Decompress an existing 7z file with original directory structure:

```
7za x compressed.7z
```

- Compress to zip format:

```
7za a -tzip compressed.zip directory_or_file_to_compress
```

- Create multipart 7zip file; `part_size` specifies part size in Bytes, Kilobytes, Megabytes or Gigabytes:

```
7za -vpart_size[b|k|m|g] compressed.7z directory_or_file_to_compress
```

ab

Apache Benchmarking tool. The simplest tool to perform a load testing.

- Execute 100 HTTP GET requests to given URL:

```
ab -n 100 url
```

- Execute 100 HTTP GET requests, processing up to 10 requests concurrently, to given URL:

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

ack

A search tool like grep, optimized for programmers.

- Find files containing “foo”:

```
ack foo
```

- Find files in a specific language:

```
ack --ruby each_with_object
```

- Count the total number of matches for the term “foo”:

```
ack -ch foo
```

- Show the file names containing “foo” and number of matches in each file:

```
ack -cl foo
```

Android Debug Bridge

Communicate with an Android emulator instance or connected Android devices.

- Check whether the adb server process is running and start it:

```
adb start-server
```

- Terminate the adb server process:

```
adb kill-server
```

- Start a remote shell in the target emulator/device instance:

```
adb shell
```

- Push an Android application to an emulator/device:

```
adb install -r apk.path
```

ag

The Silver Searcher. Like ack, but faster.

- Find files containing “foo”:

  ```
  ag foo
  ```

- Find files containing “FOO” case-insensitively:

  ```
  ag -i FOO
  ```

- Find “foo” in files with a name matching “bar”:

  ```
  ag foo -G bar
  ```

- Find files whose contents match a regular expression:

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

- Find files with a name matching “foo”:

  ```
  ag -g foo
  ```

alias

Creates an alias for a word when used. As the first word of a command.

- Create a generic alias:

  ```
  alias word="command"
  ```

- Remove an aliased command:

  ```
  unalias word
  ```

- Full list of aliased words:

  ```
  alias -p
  ```

- Turn rm an interative command:

  ```
  alias rm="rm -i"
  ```

- Override la as ls -a:

  ```
  alias la="ls -a"
  ```

apm

Atom editor Package Manager. See `atom`.

- Install packages from http://atom.io/packages and themes from http://atom.io/themes:

```
apm install package_name
```

- Remove packages/themes:

```
apm remove package_name
```

- Upgrade packages/themes:

```
apm upgrade package_name
```

apropos

Search in manpages. For example to find a new command.

- Search for keyword:

```
apropos regular_expression
```

- Search without restricting output to terminal width:

```
apropos -l regular_expression
```

ar

Create, modify, and extract from archives (.a .so .o).

- Extract all members from an archive:

```
ar -x libfoo.a
```

- List the members of an archive:

```
ar -t libfoo.a
```

- Replace or add files to an archive:

```
ar -r libfoo.a foo.o bar.o baz.o
```

- Insert an object file index (equivalent to using `ranlib`):

```
ar -s libfoo.a
```

- Create an archive with files and an accompanying object file index:

```
ar -rs libfoo.a foo.o bar.o baz.o
```

aria2c

Fast download utility. Supports HTTP(S), FTP, SFTP, BitTorrent, and Metalink.

- Download a URI to a file:

```
aria2c url
```

- Download from multiple sources:

```
aria2c url_1 url_2
```

- Download the URIs listed in a file:

```
aria2c -i filename
```

- Download with multiple connections:

```
aria2c -s connections_num url
```

- FTP download with username and password:

```
aria2c --ftp-user=username --ftp-passwd=password url
```

arp

Show and manipulate your system’s ARP cache.

- Show current arp table:

```
arp -a
```

- Clear the entire cache:

```
sudo arp -a -d
```

- Delete a specific entry:

```
arp -d address
```

- Create an entry:

```
arp -s address mac address
```

atom

A cross-platform pluggable text editor. Plugins are managed by `apm`.

- Open a file or folder:

```
atom path/to/file/or/folder
```

- Open a file or folder in a new window:

```
atom -n path/to/file/or/folder
```

autojump

Quickly jump among the directories you visit the most. Aliases like j or jc are provided for even less typing.

- Jump to a directory that contains the given pattern:

```
j pattern
```

- Jump to a sub-directory (child) of the current directory that contains the given pattern:

```
jc pattern
```

- Remove non-existing directories from the autojump database:

```
j --purge
```

- Show the entries in the autojump database:

```
j -s
```

autossh

Runs, monitors and restarts SSH connections. Auto-reconnects to keep port forwarding tunnels up. Accepts all ssh flags.

- Open an SSH session, restarting when a monitoring port fails return data:

```
autossh -M monitor_port ssh_command
```

- Open an SSH session which forwards a local port to a remote one, restarting if necessary:

```
autossh -M monitor_port -L local_port:localhost:remote_port user@host
```

- Fork before executing ssh (runs in the background) and don't open a remote shell:

```
autossh -f -M monitor_port -N ssh_command
```

- Run autossh in the background, with no monitoring port, instead relying on SSH keep-alives every 10 seconds to detect failure:

```
autossh -f -M 0 -N -o "ServerAliveInterval 10" -o "ServerAliveCountMax 3"  ssh_command
```

- Run autossh in the background, with no monitoring port, no remote shell, exiting if the port forward fails:

```
autossh -f -M 0 -N -o "ServerAliveInterval 10" -o "ServerAliveCountMax 3" -o
ExitOnForwardFailure=yes -L local_port:localhost:remote_port user@host
```

- Run autossh in the background with debug output logged to a file and ssh verbose output logged to a second file:

```
AUTOSSH_DEBUG=1 AUTOSSH_LOGFILE=log_file autossh -f -M monitor_port -v -E ssh_logfile
ssh_command
```

awk

A versatile programming language for working on files.

- Print the fifth column in a space separated file:

```
awk '{print $5}' filename
```

- Print the second column of the lines containing "something" in a space separated file:

```
awk '/something/ {print $2}' filename
```

- Print the third column in a comma separated file:

```
awk -F ',' '{print $3}' filename
```

- Sum the values in the first column and print the total:

```
awk '{s+=$1} END {print s}' filename
```

- Sum the values in the first column and pretty-print the values and then the total:

```
awk '{s+=$1; print $1} END {print "--------"; print s}' filename
```

axel

Download accelerator. Supports HTTP, HTTPS, and FTP.

- Download a URL to a file:

```
axel url
```

- Download and specify filename:

```
axel url -o filename
```

- Download with multiple connections:

```
axel -n connections_num url
```

- Search for mirrors:

```
axel -S mirrors_num url
```

- Limit download speed (bytes per second):

```
axel -s speed url
```

base32

Encode or decode file or standard input, to standard output.

- Encode a file:

```
base32 filename
```

- Decode a file:

```
base32 -d filename
```

- Encode from stdin:

```
somecommand | base32
```

- Decode from stdin:

```
somecommand | base32 -d
```

base64

Encode or decode file or standard input, to standard output.

- Encode a file:

```
base64 filename
```

- Decode a file:

```
base64 -d filename
```

- Encode from stdin:

```
somecommand | base64
```

- Decode from stdin:

```
somecommand | base64 -d
```

basename

Returns non-directory portion of a pathname.

- Show only the file name from a path:

```
basename path/to/file
```

- Show only the file name from a path, with a suffix removed:

```
basename path/to/file suffix
```

bash

Bourne-Again SHell. sh-compatible command line interpreter.

- Start interactive shell:

```
bash
```

- Execute a command:

```
bash -c command
```

- Run commands from a file:

```
bash file.sh
```

- Run commands from STDIN:

```
bash -s
```

bashmarks

Save and jump to commonly used directories using 1 character commands.

- List available bookmarks:

```
l
```

- Save the current folder as “bookmark_name”:

```
s bookmark_name
```

- Go to a bookmarked folder:

```
g bookmark_name
```

- Print a bookmarked folder’s contents:

```
p bookmark_name
```

- Delete a bookmark:

```
d bookmark_name
```

bc

Calculator.

- Run calculator in interactive mode:

```
bc -i
```

- Calculate the result of an expression:

```
bc <<< "(1 + 2) * 2 ^ 2"
```

- Calculate with the given precision:

```
bc <<< "scale=10; 5 / 3"
```

- Calculate expression with sine and cosine using mathlib:

```
bc -l <<< "s(1) + c(1)"
```

bmaptool

Create or Copy blockmaps intelligently (and therefore faster than cp or dd).

- Create a blockmap from image file:

```
bmaptool create -o blockmap.bmap source.img
```

- Copy an image file into sdb:

```
bmaptool copy --bmap blockmap.bmap source.img /dev/sdb
```

- Copy a compressed image file into sdb:

```
bmaptool copy --bmap blockmap.bmap source.img.gz /dev/sdb
```

- Copy an image file into sdb without using a blockmap:

```
bmaptool copy --nobmap source.img /dev/sdb
```

bower

A package manager optimized for front-end web development. A package can be a GitHub user/repo shorthand, a Git endpoint, a URL or a registered package.

- Install a project's dependencies, listed in its bower.json:

```
bower install
```

- Install one or more packages to the bower_components directory:

```
bower install package package
```

- Uninstall packages locally from the bower_components directory:

```
bower uninstall package package
```

- List local packages and possible updates:

```
bower list
```

- Display help information about a bower command:

```
bower help command
```

- Create a bower.json file for your package:

```
bower init
```

- Install a specific dependency version, and add it to bower.json:

```
bower install local_name=package#version --save
```

bundle

Dependency manager for the Ruby programming language.

- Install all gems defined in the gemfile expected in the working directory:

```
bundle install
```

- Update all gems by the rules defined in the gemfile and regenerate gemfile.lock:

```
bundle update
```

- Update one specific gem defined in the gemfile:

```
bundle update --source gemname
```

- Create a new gem skeleton:

```
bundle gem gemname
```

c99

Compiles C programs according to the ISO C standard.

- Compile source file(s) and create an executable:

```
c99 file.c
```

- Compile source file(s) and create an executable with a custom name:

```
c99 -o executable_name file.c
```

- Compile source file(s) and create object file(s):

```
c99 -c file.c
```

- Compile source file(s), link with object file(s), and create an executable:

```
c99 file.c file.o
```

cal

Prints calendar information.

- Display a calendar for the current month:

```
cal
```

- Display a calendar for a specific month:

```
cal -m month_number
```

- Display a 12 month calendar for the current year:

```
cal -y
```

- Display a 12 month calendar for a specific year:

```
cal 2016
```

- Display date of Easter (western churches):

```
ncal -e year
```

calibre-server

A server application that can be used to distribute ebooks over a network. Ebooks must be imported into the library using the GUI or calibredb before. Part of the Calibre ebook library.

- Start a server to distribute ebooks. Access at http://localhost:8080:

```
calibre-server
```

- Start server on different port. Access at http://localhost:port:

```
calibre-server --port port
```

- Password protect the server:

```
calibre-server --username username --password password
```

calibredb

Tool to manipulate the your ebook database. Part of the Calibre ebook library.

- List ebooks in the library with additional information:

```
calibredb list
```

- Search for ebooks displaying additional information:

```
calibredb list --search search-term
```

- Search for just ids of ebooks:

```
calibredb search search term
```

- Add one or more ebooks to the library:

```
calibredb add file1 file2 …
```

- Remove one or more ebooks from the library. You need ebook-ids (see above):

```
calibredb remove id1 id2 …
```

cat

Print and concatenate files.

- Print the contents of a file to the standard output:

```
cat file
```

- Concatenate several files into the target file:

```
cat file1 file2 > target-file
```

- Append several files into the target file:

```
cat file1 file2 >> target-file
```

- Number all output lines:

```
cat -n file
```

cd

Change the current working directory.

- Go to the given directory:

```
cd /path/to/directory
```

- Go to home directory of current user:

```
cd
```

- Go up to the parent of the current directory:

```
cd ..
```

- Go to the previously chosen directory:

```
cd -
```

chgrp

Change group ownership of files and folders.

- Change the owner of a file/folder:

```
chgrp group path/to/file
```

- Recursively change the owner of a folder and its contents:

```
chgrp -R group path/to/folder
```

- Change the owner of a symbolic link:

```
chgrp -h user path/to/symlink
```

- Change the owner of a file/folder to match a reference file:

```
chgrp --reference=path/to/reference_file path/to/file
```

chmod

Change the access permissions of a file or directory.

- Give the (u)ser who owns a file the right to e(x)ecute it:

```
chmod u+x file
```

- Give the user rights to (r)ead and (w)rite to a file/directory:

```
chmod u+rw file
```

- Remove executable rights from the (g)roup:

```
chmod g-x file
```

- Give (a)ll users rights to read and execute:

```
chmod a+rx file
```

- Give (o)thers (not in the file owner's group) the same rights as the group:

```
chmod o=g file
```

chown

Change user and group ownership of files and folders.

- Change the owner user of a file/folder:

```
chown user path/to/file
```

- Change the owner user and group of a file/folder:

```
chown user:group path/to/file
```

- Recursively change the owner of a folder and its contents:

```
chown -R user path/to/folder
```

- Change the owner of a symbolic link:

```
chown -h user path/to/symlink
```

- Change the owner of a file/folder to match a reference file:

```
chown --reference=path/to/reference_file path/to/file
```

chsh

Change user’s login shell.

- Change shell:

```
chsh -s path/to/shell_binary username
```

cksum

Calculates CRC checksums and byte counts of a file. Note, on old UNIX systems the CRC implementation may differ.

- Display a 32 bit checksum, size in bytes and filename:

```
cksum filename
```

clang

Compiler for C, C++, and Objective-C source files. Can be used as a drop-in replacement for GCC.

- Compile a source code file into an executable binary:

```
clang input_source.c -o output_executable
```

- Activate output of all errors and warnings:

```
clang input_source.c -Wall -o output_executable
```

- Include libraries located at a different path than the source file:

```
clang input_source.c -o output_executable -Iheader_path -Llibrary_path -llibrary_name
```

cmp

Compare two files.

- Find the byte number and line number of the first difference between the files:

```
cmp file1 file2
```

- Find the byte number and differing bytes of every difference:

```
cmp -l file1 file2
```

comm

Select or reject lines common to two files. Both files must be sorted.

- Produce three tab-separated columns: lines only in first file, lines only in second file and common lines:

```
comm file1 file2
```

- Print only lines common to both files:

```
comm -12 file1 file2
```

- Print only lines common to both files, read one file from stdin:

```
cat file1 | comm -12 - file2
```

- Print lines only found in first file:

```
comm -23 file1 file2
```

- Print lines only found in second file:

```
comm -13 file1 file2
```

convert

Imagemagick image conversion tool.

- Convert an image from JPG to PNG:

```
convert image.jpg image.png
```

- Scale an image 50% it's original size:

```
convert image.png -resize 50% image2.png
```

- Scale an image keeping the original aspect ratio to a maximum dimension of 640x480:

```
convert image.png -resize 640x480 image2.png
```

- Horizontally append images:

```
convert image1.png image2.png image3.png +append image123.png
```

convmv

Convert filenames (NOT file content) from one encoding to another.

- Test filename encoding conversion (don't actually change the filename):

  ```
  convmv -f from_encoding -t to_encoding input_file
  ```

- Convert filename encoding and rename the file to the new enconding:

  ```
  convmv -f from_encoding -t to_encoding --notest input_file
  ```

cordova

Mobile apps with HTML, CSS & JS.

- Create a cordova project:

  ```
  cordova create path package.name project.name
  ```

- Display the current workspace status:

  ```
  cordova info
  ```

- Add a cordova platform:

  ```
  cordova platform add platform
  ```

- Remove a cordova platform:

  ```
  cordova platform remove platform
  ```

- Add a cordova plugin:

  ```
  cordova plugin add pluginid
  ```

- Remove a cordova plugin:

  ```
  cordova plugin remove pluginid
  ```

cowsay

Generate an ASCII character like a cow or sheep saying or thinking something.

- Print an ASCII cow saying "Hello world!":

```
cowsay "Hello world!"
```

- Print an ASCII dragon saying "Hello!":

```
echo "Hello!" | cowsay -f dragon
```

- Print a stoned thinking ASCII cow:

```
cowthink -s "I'm just a cow, not a great thinker ..."
```

cp

Copy files.

- Copy files in arbitrary locations:

```
cp /path/to/original /path/to/copy
```

- Copy a file to a parent directory:

```
cp /path/to/original ../path/to/copy
```

- Copy directories recursive using the option -r:

```
cp -r /path/to/original /path/to/copy
```

- Show files as they are copied:

```
cp -vr /path/to/original /path/to/copy
```

- Make a copy of a file, adding an extension:

```
cp file.html{,.backup}
```

- Make a copy of a file, changing the extension:

```
cp file.{html,backup}
```

crontab

Schedule cron jobs to run on a time interval for the current user.

- Edit the crontab file for the current user:

  ```
  crontab -e
  ```

- View a list of existing cron jobs for current user:

  ```
  crontab -l
  ```

- Remove all cron jobs for the current user:

  ```
  crontab -r
  ```

csvclean

Finds and cleans common syntax errors in CSV files. Included in csvkit.

- Clean a CSV file:

  ```
  csvclean bad.csv
  ```

- List locations of syntax errors in a CSV file:

  ```
  csvclean -n bad.csv
  ```

csvcut

Filter and truncate CSV files. Like Unix's `cut` command, but for tabular data. Included in csvkit.

- Print indices and names of all columns:

  ```
  csvcut -n data.csv
  ```

- Extract the first and third columns:

  ```
  csvcut -c 1,3 data.csv
  ```

- Extract all columns **except** the fourth one:

  ```
  csvcut -C 4 data.csv
  ```

- Extract the columns named "id" and "first name" (in that order):

  ```
  csvcut -c id,"first name" data.csv
  ```

csvformat

Convert a CSV file to a custom output format. Included in csvkit.

– Convert to a tab-delimited file (TSV):

```
csvformat -T data.csv
```

– Convert delimiters to a custom character:

```
csvformat -D "custom_character" data.csv
```

– Convert line endings to carriage return (ˆM) + line feed:

```
csvformat -M "\r\n" data.csv
```

– Minimize use of quote characters:

```
csvformat -U 0 data.csv
```

– Maximize use of quote characters:

```
csvformat -U 1 data.csv
```

csvgrep

Filter CSV rows with string and pattern matching. Included in csvkit.

– Find rows that have a certain string in column 1:

```
csvgrep -c 1 -m string_to_match data.csv
```

– Find rows in which columns 3 or 4 match a certain regex pattern:

```
csvgrep -c 3,4 -r regex_pattern data.csv
```

– Find rows in which the "name" column does NOT include the string "John Doe":

```
csvgrep -i -c name -m "John Doe" data.csv
```

csvlook

Render a CSV file in the console as a fixed-width table. Included in csvkit.

– View a CSV file:

```
csvlook data.csv
```

csvpy

Loads a CSV file into a Python shell. Included in csvkit.

- Load a CSV file into a `CSVKitReader` object:

  ```
  csvpy data.csv
  ```

- Load a CSV file into a `CSVKitDictReader` object:

  ```
  csvpy --dict data.csv
  ```

csvsort

Sorts CSV files. Included in csvkit.

- Sort a CSV file by column 9:

  ```
  csvsort -c 9 data.csv
  ```

- Sort a CSV file by the "name" column in descending order:

  ```
  csvsort -r -c name data.csv
  ```

- Sort a CSV file by column 2, then by column 4:

  ```
  csvsort -c 2,4 data.csv
  ```

- Sort a CSV file without inferring data types:

  ```
  csvsort --no-inference -c columns data.csv
  ```

csvstat

Print descriptive statistics for all columns in a CSV file. Included in csvkit.

- Show all stats for all columns:

  ```
  csvstat data.csv
  ```

- Show all stats for columns 2 and 4:

  ```
  csvstat -c 2,4 data.csv
  ```

- Show sums for all columns:

  ```
  csvstat --sum data.csv
  ```

- Show the max value length for column 3:

  ```
  csvstat -c 3 --len data.csv
  ```

- Show the number of unique values in the "name" column:

  ```
  csvstat -c name --unique data.csv
  ```

curl

Transfers data from or to a server. Supports most protocols including HTTP, FTP, POP.

– Download a URL to a file:

```
curl "URL" -o filename
```

– Send form-encoded data:

```
curl --data name=bob http://localhost/form
```

– Send JSON data:

```
curl -X POST -H "Content-Type: application/json" -d '{"name":"bob"}' http://
localhost/login
```

– Specify an HTTP method:

```
curl -X DELETE http://localhost/item/123
```

– Head request:

```
curl --head http://localhost
```

– Include an extra header:

```
curl -H "X-MyHeader: 123" http://localhost
```

– Pass a user name and password for server authentication:

```
curl -u myusername:mypassword http://localhost
```

– Pass client certificate and key for a secure resource:

```
curl -v –key key.pem –cacert ca.pem –cert client.pem -k https://localhost
```

cut

Cut out fields from STDIN or files.

- Cut out the first sixteen characters of each line of STDIN:

```
cut -c 1-16
```

- Cut out the first sixteen characters of each line of the given files:

```
cut -c 1-16 file
```

- Cut out everything from the 3rd character to the end of each line:

```
cut -c3-
```

- Cut out the fifth field, split on the colon character of each line:

```
cut -d':' -f5
```

- Cut out the fields five and 10, split on the colon character of each line:

```
cut -d':' -f5,10
```

- Cut out the fields five through 10, split on the colon character of each line:

```
cut -d':' -f5-10
```

date

Set or display the system date.

- Display the date using the default locale:

```
date +"%c"
```

- Display the date in UTC and ISO 8601 format:

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

deluser

Remove a user account or remove a user from a group.

- Remove a user:

```
deluser name
```

- Remove a user along with their home directory and mail spool:

```
deluser -r name
```

- Remove a user from a group:

```
deluser name group
```

df

Gives an overview of the file system disk space usage.

- Display all file systems and their disk usage:

```
df
```

- Display all file systems and their disk usage in human readable form:

```
df -h
```

dhcpwn

Test DHCP IP exhaustion attacks and sniff local DHCP traffic.

- Flood the network with IP requests:

```
dhcpwn --interface network_interface flood --count number_of_requests
```

- Sniff local DHCP traffic:

```
dhcpwn --interface network_interface sniff
```

diff

Compare files and directories.

- Compare files:

```
diff file1 file2
```

- Compare files, ignoring white spaces:

```
diff -w file1 file2
```

- Compare files, showing differences side by side:

```
diff -y file1 file2
```

- Compare directories recursively:

```
diff -r directory1 directory2
```

- Compare directories, only showing the names of files that differ:

```
diff -rq directory1 directory2
```

dig

DNS Lookup utility.

- Lookup the IP(s) associated with a hostname (A records):

```
dig +short hostname.com
```

- Lookup the mail server associated with a given domain name (MX record):

```
dig +short hostname.com MX
```

- Specify an alternate DNS server to query (8.8.8.8 is google's public DNS):

```
dig @8.8.8.8 hostname.com
```

dirs

Displays or manipulates the directory stack. The directory stack is a list of recently visited directories that can be manipulated with the pushd and popd commands.

- Display the directory stack with a space between each entry:

```
dirs
```

- Display the directory stack with one entry per line:

```
dirs -p
```

- Display only the nth entry in the directory stack, starting at 0:

```
dirs +N
```

- Clear the directory stack:

```
dirs -c
```

docker

Manage Docker containers and images.

- List of running docker containers:

```
docker ps
```

- List all docker containers (running and stopped):

```
docker ps -a
```

- Start a container:

```
docker start container
```

- Stop a container:

```
docker stop container
```

- Start a container from an image and get a shell inside of it:

```
docker run -it image bash
```

- Run a command inside of an already running container:

```
docker exec container command
```

Dokku

Docker powered mini-Heroku (PaaS). Easily deploy multiple apps to your server in different languages using a single `git-push` command.

- List runinng apps:

```
dokku apps
```

- Create an app:

```
dokku apps:create app_name
```

- Remove an app:

```
dokku apps:destroy app_name
```

- Install plugin:

```
dokku plugin:install full_repo_url
```

- Link database to an app:

```
dokku db:link db_name app_name
```

drush

A command-line shell and scripting interface for Drupal.

- Download module "foo":

```
drush dl foo
```

- Download version 7.x-2.1-beta1 of module "foo":

```
drush dl foo-7.x-2.1-beta1
```

- Enable module "foo":

```
drush en foo
```

- Disable module "foo":

```
drush dis foo
```

- Clear all caches:

```
drush cc all
```

- Clear CSS and JavaScript caches:

```
drush cc css-js
```

ebook-convert

Can be used to convert ebooks between common formats, e.g., pdf, epub and mobi. Part of the Calibre ebook library tool.

- Convert an ebook into another format:

```
ebook-convert source destination
```

echo

Print given arguments.

- Print a text message. Note: quotes are optional:

```
echo "Hello World"
```

- Print a message with environment variables:

```
echo "My path is $PATH"
```

electrum

Ergonomic Bitcoin wallet and private key management.

- Create a new wallet:

```
electrum -w new-wallet.dat create
```

- Restore an existing wallet from seed offline:

```
electrum -w recovery-wallet.dat restore -o
```

- Create a signed transaction offline:

```
electrum mktx recipient amount -f 0.0000001 -F from -o
```

- Display all wallet receiving addresses:

```
electrum listaddresses -a
```

- Sign a message:

```
electrum signmessage address message
```

- Verify a message:

```
electrum verifymessage address signature message
```

- Connect only to a specific electrum-server instance:

```
electrum -p socks5:127.0.0.1:9050 -s 56ckl5obj37gypcu.onion:50001:t -1
```

emacs

The extensible, customizable, self-documenting, real-time display editor.

– Open emacs in console mode (without X window):

```
emacs -nw
```

– Open a file in emacs:

```
emacs filename
```

– Exit emacs:

```
C-x C-c
```

enca

Detect and convert encoding of text files.

– Detect file(s) encoding according to your system's locale:

```
enca file(s)
```

– Detect file(s) encoding; -L option tells enca the current language; language is in the POSIX/C locale format, e.g. zh_CN, en_US etc:

```
enca -L language file(s)
```

– Convert file(s) to specified encoding:

```
enca -L language -x to_encoding file(s)
```

– Save original_file as new_file and convert new_file to specified encoding:

```
enca -L language -x to_encoding < original_file > new_file
```

env

Show the environment or run a program in a modified environment.

- Show the environment:

```
env
```

- Run a program. Often used in scripts after the shebang (#!) for looking up the path to the program:

```
env program
```

- Clear the environment and run a program:

```
env -i program
```

- Remove variable from the environment and run a program:

```
env -u variable program
```

- Set a variable and run a program:

```
env variable=value program
```

espeak

Uses text-to-speech to speak through the default sound device.

- Speak a phrase aloud:

```
espeak "I like to ride my bike."
```

- Speak a file aloud:

```
espeak -f filename
```

- Save output to a WAV audio file, rather than speaking it directly:

```
espeak -w filename.wav "It's GNU plus Linux"
```

- Use a different voice:

```
espeak -v voice
```

exiftool

Read and write meta information in files.

- Remove all EXIF metadata from the given files:

```
exiftool -All= file
```

- Increase time photo taken by 1 hour in directory:

```
exiftool "-AllDates+=0:0:0 1:0:0" directory
```

- Decrease time photo taken by 1 day and 2 hours on JPEGs only:

```
exiftool "-AllDates-=0:0:1 2:0:0" -ext jpg
```

- Change only DateTimeOriginal by -1.5 hours & do not keep backups:

```
exiftool -DateTimeOriginal-=1.5 -overwrite_original
```

- Rename all JPEGs according to a DateTimeOriginal recursively:

```
exiftool '-filename<DateTimeOriginal' -d %Y-%m-%d_%H-%M-%S%%lc.%%e directory
-r -ext jpg
```

fdupes

Finds duplicate files in a given. Set of directories.

- Search a single directory:

```
fdupes directory
```

- Search multiple directories:

```
fdupes directory1 directory2
```

- Search all directories recursively:

```
fdupes -r directory
```

- Search multiple directories, one recursively:

```
fdupes directory1 -R directory2
```

ffmpeg

Video conversion tool.

- Extract the sound from a video and save it as MP3:

```
ffmpeg -i video-filename -vn -ar 44100 -ac 2 -ab 192 -f mp3 sound.mp3
```

- Convert frames from a video into individual numbered images:

```
ffmpeg -i video-filename image%d.png
```

- Combine numbered images (image1.jpg, image2.jpg, etc) into a video:

```
ffmpeg -f image2 -i image%d.jpg video.mpg
```

- Convert AVI video to MP4. AAC Audio @ 128kbit, Video @ 1250Kbit:

```
ffmpeg -i in.avi -acodec libfaac -ab 128k -vcodec mpeg4 -b 1250K out.mp4
```

fg

Run jobs in foreground.

- Bring most recently suspended background job to foreground:

```
fg
```

- Bring a specific job to foreground:

```
fg job_id
```

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
```

find

Find files under the given directory tree, recursively.

- Find files by extension:

```
find root_path -name '*.py'
```

- Find files matching path pattern:

```
find root_path -path '**/lib/**/*.py'
```

- Run a command for each file, use {} within the command to access the filename:

```
find root_path -name '*.py' -exec wc -l {} \;
```

- Find files modified since a certain time:

```
find root_path -name '*.py' -mtime -1d
```

- Find files using case insensitive name matching, of a certain size:

```
find root_path -size +500k -size -10MB -iname '*.TaR.gZ'
```

- Delete files by name, older than a certain number of days:

```
find root_path -name '*.py' -mtime -180d -delete
```

- Find empty files or directories:

```
find root_path -empty
```

- Find files matching more than one search criteria:

```
find root_path -name '*.py' -or -name '*.r'
```

for

Shell loop over parameters.

- Perform a command with different arguments:

```
for argument in 1 2 3; do command $argument; done
```

- Perform a command in every directory:

```
for d in *; do (cd $d; command); done
```

fortune

Print a random quotation (fortune-cookie style).

- Print a quotation:

```
fortune
```

- Print an offensive quotation:

```
fortune -o
```

- Print a long quotation:

```
fortune -l
```

- Print a short quotation:

```
fortune -s
```

- List the available quotation database files:

```
fortune -f
```

- Print a quotation from one of the database files listed by `fortune -f`:

```
fortune filename
```

fswebcam

Small and simple webcam for *nix.

- Take a picture:

```
fswebcam filename
```

- Take a picture with custom resolution:

```
fswebcam -r widthxheight filename
```

- Take a picture from selected device(Default is /dev/vidoe0):

```
fswebcam -d device filename
```

- Take a picture with timestamp(timestamp string is formatted by strftime):

```
fswebcam --timestamp timestamp filename
```

fzf

Command line fuzzy finder.

- Start finder on all files from arbitrary locations:

  ```
  find path/to/search -type f | fzf
  ```

- Start finder on running processes:

  ```
  ps axu | fzf
  ```

- Select mutliple files with `Shift-TAB` and write to a file:

  ```
  find path/to/search_files -type f | fzf -m > filename
  ```

- Start finder with a given query:

  ```
  fzf -q "query"
  ```

- Start finder on entries that start with core and end with either go, rb, or py:

  ```
  fzf -q "^core go$ | rb$ | py$"
  ```

- Start finder on entries that not match pyc and match exactly travis:

  ```
  fzf -q "!pyc 'travis"
  ```

gcc

Preprocesses and compiles C and C++ source files, then assembles and links them together.

- Compile multiple source files into executable:

  ```
  gcc source1.c source2.c -o executable
  ```

- Allow warnings, debug symbols in output:

  ```
  gcc source.c -Wall -Og -o executable
  ```

- Include libraries from a different path:

  ```
  gcc source.c -o executable -Iheader_path -Llibrary_path -llibrary_name
  ```

- Compile source code into Assembler instructions:

  ```
  gcc -S source.c
  ```

gdb

The GNU Debugger.

- Debug an executable:

```
gdb executable
```

- Attach a process to gdb:

```
gdb -p procID
```

- Execute given GDB commands upon start:

```
gdb -ex "commands" executable
```

- Start gdb and pass arguments:

```
gdb --args executable argument1 argument2
```

gem

Interact with the package manager for the Ruby programming language.

- Install latest version of a gem:

```
gem install gemname
```

- Install specific version of a gem:

```
gem install gemname -v 1.0.0
```

- Update a gem:

```
gem update gemname
```

- List all gems:

```
gem list
```

- Uninstall a gem:

```
gem uninstall gemname
```

gifsicle

Create gifs.

- Make a GIF animation with gifsicle:

  ```
  gifsicle --delay=10 --loop *.gif > anim.gif
  ```

- Extract frames from an animation:

  ```
  gifsicle anim.gif '#0' > firstframe.gif
  ```

- You can also edit animations by replacing, deleting, or inserting frames:

  ```
  gifsicle -b anim.gif --replace '#0' new.gif
  ```

git add

Adds changed files to the index.

- Add a file to the index:

  ```
  git add path/to/file
  ```

- Add all files (tracked and untracked):

  ```
  git add .
  ```

- Only add already tracked files:

  ```
  git add -u
  ```

- Also add ignored files:

  ```
  git add -f
  ```

- Add parts of a file interactively:

  ```
  git add -p path/to/file
  ```

git blame

Show commit hash and last author on each line of a file.

- Print file with author name and commit hash on each line:

  ```
  git blame file
  ```

- Print file with author email and commit hash on each line:

  ```
  git blame -e file
  ```

git branch

Main command for working with branches.

- List local branches. The current branch is highlighted by *:

```
git branch
```

- List all local and remote branches:

```
git branch -a
```

- Create new branch based on current branch:

```
git branch BRANCH-NAME
```

- Delete a local branch:

```
git branch -d BRANCH-NAME
```

- Move/Rename a branch:

```
git branch -m
```

git checkout

Checkout a branch or paths to the working tree.

- Switch to another branch:

```
git checkout BRANCH-NAME
```

- Create and switch to a new branch:

```
git checkout -b BRANCH-NAME
```

- Undo unstaged local modification:

```
git checkout .
```

git clone

Clone an existing repository.

- Clone an existing repository:

```
git clone REMOTE-REPOSITORY-LOCATION
```

- For cloning from the local machine:

```
git clone -l
```

- Do it quietly:

```
git clone -q
```

- Clone an existing repository, and truncate to the specified number of revisions, save your time mostly:

```
git clone --depth 10 REMOTE-REPOSITORY-LOCATION
```

git commit

Commit staged files to the repository.

- Commit staged files to the repository with comment:

```
git commit -m MESSAGE
```

- Replace the last commit with currently staged changes:

```
git commit --amend
```

git config

Get and set repository or global options.

- Print list of options for current repository:

```
git config --list --local
```

- Print global list of options, set in ~/.gitconfig:

```
git config --list --global
```

- Get full list of options:

  ```
  git config --list
  ```

- Get value of alias.ls option:

  ```
  git config alias.st
  ```

- Set option alias.ls=status in file ~/.gitconfig:

  ```
  git config --global alias.ls "status"
  ```

- Remove option alias.st from ~/.gitconfig:

  ```
  git config --global --unset alias.st
  ```

git diff

Show changes to tracked files.

- Show changes to tracked files:

  ```
  git diff PATHSPEC
  ```

- Show only names of changed files:

  ```
  git diff --name-only PATHSPEC
  ```

- Output a condensed summary of extended header information:

  ```
  git diff --summary PATHSPEC
  ```

- Show staged (added, but not yet committed) changes only:

  ```
  git diff --staged
  ```

git fetch

Download objects and refs from a remote repository.

- Fetch new branches and update remote-tracking branches:

  ```
  git fetch remote_name
  ```

- Fetch the latest changes from all remote git servers:

  ```
  git fetch --all
  ```

git init

Initializes a new local Git repository.

– Initialize a new local repository:

```
git init
```

– Initialize a barebones repository, suitable for use as a remote over ssh:

```
git init --bare
```

git log

Show a history of commits.

– Show a history of commits:

```
git log
```

– Show the history of a particular file or directory, including differences:

```
git log -p path
```

– Show only the first line of each commits:

```
git log --oneline
```

git merge

Merge branches.

– Merge a branch with your current branch:

```
git merge BRANCH-NAME
```

– Edit the merge message:

```
git merge -e BRANCH-NAME
```

git mv

Move or rename files and update the git index.

- Move file inside the repo and add the movement to the next commit:

  ```
  git mv path/to/file new/path/to/file
  ```

- Rename file and add renaming to the next commit:

  ```
  git mv filename new_filename
  ```

- Overwrite the file in the target path if it exists:

  ```
  git mv --force file target
  ```

git pull

Fetch branch from a remote repository and merge it to local repository.

- Download changes from default remote repository and merge it:

  ```
  git pull
  ```

- Download changes from default remote repository and use fast forward:

  ```
  git pull --rebase
  ```

- Download changes from given remote repository and branch, then merge them into HEAD:

  ```
  git pull remote_name branch
  ```

git push

Push commits to a remote repository.

- Publish local changes on a remote branch:

  ```
  git push REMOTE-NAME LOCAL-BRANCH
  ```

- Publish local changes on a remote branch of different name:

  ```
  git push REMOTE-NAME LOCAL-BRANCH:REMOTE-BRANCH
  ```

- Remove remote branch:

  ```
  git push REMOTE-NAME :REMOTE-BRANCH
  ```

- Remove remote branches which don't exist locally:

  ```
  git push --prune REMOTE-NAME
  ```

- Publish tags:

  ```
  git push --tags
  ```

git rebase

Apply local commits on top of another branch’s history.

- Rebase your local branch interactively with the latest changes in local master:

```
git rebase -i master
```

- Rebase your local branch interactively with the latest changes from upstream:

```
git fetch origin; git rebase -i origin/master
```

- Handle an active rebase merge failure, after editing conflicting file(s):

```
git rebase --continue
```

- Abort a rebase in-progress:

```
git rebase --abort
```

git remote

Manage set of tracked repositories (“remotes”).

- Show a list of existing remotes, their names and URL:

```
git remote -v
```

- Add a remote:

```
git remote add remote_name remote_url
```

- Change the URL of a remote:

```
git remote set-url remote_name new_url
```

- Remove a remote:

```
git remote remove remote_name
```

- Rename a remote:

```
git remote rename old_name new_name
```

git rm

Remove files from repository index and local filesystem.

- Remove file from repository index and filesystem:

```
git rm file
```

- Remove directory:

```
git rm -r directory
```

- Remove file from repository index but keep it untouched locally:

```
git rm --cached file
```

git stash

Stash local Git changes in a temporary area.

- Stash current changes (except new files):

```
git stash save optional_stash_name
```

- Include new files in the stash (leaves the index completely clean):

```
git stash save -u optional_stash_name
```

- List all stashes:

```
git stash list
```

- Re-apply the latest stash:

```
git stash pop
```

- Re-apply a stash by name:

```
git stash apply stash_name
```

- Drop a stash by an index:

```
git stash drop stash@{index}
```

git status

Show the index (changed files).

- Show changed files which are not yet added for commit:

```
git status
```

- Give output in short format:

```
git status -s
```

git submodule

Inspects, updates and manages submodules.

- Install a repository's specified submodules:

```
git submodule update --init --recursive
```

- Add a git repository as a submodule:

```
git submodule add repository-url
```

- Update every submodule to its latest commit:

```
git submodule foreach git pull
```

git svn

Bidirectional operation between a Subversion repository and Git.

- Clone an SVN repository:

```
git svn clone http://example.com/my_subversion_repo local_dir
```

- Update local clone from the upstream SVN repository:

```
git svn rebase
```

- Commit back to SVN repository:

```
git svn dcommit
```

git tag

Create, list, delete or verify tags. Tag is reference to specific commit.

- List all tags:

```
git tag
```

- Create a tag with the given name pointing to the current commit:

```
git tag tag_name
```

- Create an annotated tag with the given message:

```
git tag tag_name -m tag_message
```

- Delete the tag with the given name:

```
git tag -d tag_name
```

git

Main command for all git commands.

- Check the Git version:

```
git --version
```

- Call general help:

```
git --help
```

- Call help on a command:

```
git help COMMAND
```

- Execute Git command:

```
git COMMAND
```

glances

A cross-platform system monitoring tool.

- Run in terminal:

```
glances
```

- Run in web server mode to show results in browser:

```
glances -w
```

- Run in server mode to allow connections from other Glances clients:

```
glances -s
```

- Connect to a Glances server:

```
glances -c hostname
```

- Require a password in (web) server mode:

```
glances -s --password
```

gpg

Gnu Privacy Guard.

- Sign doc.txt without encryption (writes output to doc.txt.asc):

```
gpg --clearsign doc.txt
```

- Encrypt doc.txt for alice@example.com (output to doc.txt.gpg):

```
gpg --encrypt --recipient alice@example.com doc.txt
```

- Encrypt doc.txt with only a passphrase (output to doc.txt.gpg):

```
gpg --symmetric doc.txt
```

- Decrypt doc.txt.gpg (output to STDOUT):

```
gpg --decrypt doc.txt.gpg
```

- Import a public key:

```
gpg --import public.gpg
```

Gradle

Gradle is the official build system for Android Studio.

- Compile a package:

  `gradle build`

- Clear the build folder:

  `gradle clean`

- Compile and Release package:

  `gradle assembleRelease`

grep

Matches patterns in input text. Supports simple patterns and regular expressions.

- Search for an exact string:

  `grep something file_path`

- Search in case-insensitive mode:

  `grep -i something path/to/file`

- Search recursively in current directory for an exact string:

  `grep -r something .`

- Use a regular expression (-E for extended regex, supporting ?, +, {}, () and |):

  `grep -e ^regex$ path/to/file`

- Print 3 lines of context around each match:

  `grep -C 3 something path/to/file`

- Print the count of matches instead of the matching text:

  `grep -c something path/to/file`

- Print line number for each match:

  `grep -n something path/to/file`

- Use the standard input instead of a file:

  `cat path/to/file | grep something`

- Invert match for excluding specific strings:

  `grep -v something`

gulp

JavaScript task runner and streaming build system. Tasks are defined within gulpfile.js at the project root.

- Run the default task:

```
gulp
```

- Run individual tasks:

```
gulp task othertask
```

gzip

Compress/uncompress files with gzip compression (LZ77).

- Compress a file, replacing it with a gzipped compressed version:

```
gzip file.ext
```

- Decompress a file, replacing it with the original uncompressed version:

```
gzip -d file.ext.gz
```

- Compress a file specifying the output filename:

```
gzip -c file.ext > compressed-file.ext.gz
```

- Uncompress a gzipped file specifying the output filename:

```
gzip -c -d file.ext.gz > uncompressed-file.ext
```

- Specify the compression level. 1=Fastest (Worst), 9=Slowest (Best), Default level is 6:

```
gzip -9 -c file.ext > compressed-file.ext.gz
```

HandBrakeCLI

Video conversion tool.

- Convert a video file to MKV (AAC 160kbit audio and x264 CRF20 video):

```
HandBrakeCLI -i input.avi -o output.mkv -e x264 -q 20 -B 160
```

- Resize a video file to 320x240:

```
HandBrakeCLI -i input.mp4 -o output.mp4} -w 320 -l 240
```

- List available presets:

```
HandBrakeCLI --preset-list
```

- Convert an AVI video to MP4 using the Android preset:

```
HandBrakeCLI --preset="Android" -i input.ext -o output.mp4
```

haxelib

Haxe Library Manager.

- Search for a Haxe library:

  `haxelib search keyword`

- Install a Haxe library:

  `haxelib install libname`

- Upgrade all installed Haxe libraries:

  `haxelib upgrade`

- Install the development version of a library from a Git repository:

  `haxelib git libname GIT-URL`

history

Command Line history.

- Display the commands history list with line numbers:

  `history`

- Clear the commands history list (only for current bash shell):

  `history -c`

- Overwrite history file with history of current bash shell (often combined with `history -c` to purge history):

  `history -w`

hn

Command-line interface for Hacker News.

- View stories on Hacker News:

  `hn`

- View *number* of stories on Hacker News:

  `hn --limit number`

- View stories on Hacker News, and keep the list open after selecting a link:

  `hn --keep-open`

- View stories on Hacker News sorted by submission date:

  `hn --latest`

host

Lookup Domain Name Server.

- Lookup A, AAAA, and MX records of a domain:

```
host domain
```

- Lookup a field (CNAME, TXT,...) of a domain:

```
host -t field domain
```

- Reverse lookup an IP:

```
host ip_address
```

http

HTTPie: HTTP client, a user-friendly cURL replacement.

- Download a URL to a file:

```
http -d example.org
```

- Send form-encoded data:

```
http -f example.org name='bob' profile-picture@'bob.png'
```

- Send JSON object:

```
http example.org name='bob'
```

- Specify an HTTP method:

```
http HEAD example.org
```

- Include an extra header:

```
http example.org X-MyHeader:123
```

- Pass a user name and password for server authentication:

```
http -a username:password example.org
```

- Specify raw request body via stdin:

```
cat data.txt | http PUT example.org
```

hub

A command-line wrapper for git that makes you better at GitHub. The commands can also be used using "git" instead of "hub".

- Clone a repository you own:

```
hub clone repo_name
```

- Clone another user repository:

```
hub clone github_username/repo_name
```

- Open the current project's issues page:

```
hub browse -- issues
```

iconv

Converts text from one encoding to another.

- Convert file and print to stdout:

```
iconv -f from_encoding -t to_encoding input_file
```

- Convert file to current locale:

```
iconv -f from_encoding input_file > output_file
```

- List supported encodings:

```
iconv -l
```

if

Simple shell conditional.

- Echo a different thing depending on a command's success:

```
command && echo "success" || echo "failure"
```

- Full if syntax:

```
if condition; then echo "true"; else echo "false"; fi
```

ifconfig

Ifconfig - Interface Configurator, used to configure network interfaces.

- View network settings of an ethernet adapter:

```
ifconfig eth0
```

- Display details of all interfaces, including disabled interfaces:

```
ifconfig -a
```

- Disable eth0 interface:

```
ifconfig eth0 down
```

- Enable eth0 interface:

```
ifconfig eth0 up
```

- Assign IP address to eth0 interface:

```
ifconfig eth0 ip_address
```

in2csv

Converts various tabular data formats into CSV. Included in csvkit.

- Convert an XLS file to CSV:

```
in2csv data.xls
```

- Convert a DBF file to a CSV file:

```
in2csv data.dbf > data.csv
```

- Convert a specific sheet from an XLSX file to CSV:

```
in2csv --sheet=sheet_name data.xlsx
```

- Pipe a JSON file to in2csv:

```
cat data.json | in2csv -f json > data.csv
```

inkscape

An SVG (Scalable Vector Graphics) editing program. Use -z to not open the GUI and only process files in the console.

- Open an SVG file in the Inkscape GUI:

```
inkscape filename.svg
```

- Export an SVG file into a bitmap with the default format (PNG) and the default resolution (90 DPI):

```
inkscape filename.svg -e filename.png
```

- Export an SVG file into a bitmap of 600x400 pixels (aspect ratio distortion may occur):

```
inkscape filename.svg -e filename.png -w 600 -h 400
```

- Export a single object, given its ID, into a bitmap:

```
inkscape filename.svg -i id -e object.png
```

- Export an SVG document to PDF, converting all texts to paths:

```
inkscape filename.svg --export-pdf==filename.pdf --export-text-to-path
```

- Duplicate the object with id="path1555", rotate the duplicate 90 degrees, save the SVG, and quit:

```
inkscape filename.svg --select=path1555 --verb=EditDuplicate --verb=ObjectRotate90
--verb=FileSave --verb=FileClose
```

ionice

Get or set program I/O scheduling class and priority. Scheduling classes: 1 (realtime), 2 (best-effort), 3 (idle). Priority levels: 0 (the highest) - 7 (the lowest).

- Set I/O scheduling class of a running process:

```
ionice -c scheduling_class -p pid
```

- Run a command with custom I/O scheduling class and priority:

```
ionice -c scheduling_class -n priority command
```

- Print the I/O scheduling class and priority of a running process:

```
ionice -p pid
```

ioping

Monitor I/O latency in real time.

– Show disk I/O latency using the default values and the current directory:

```
ioping .
```

– Measure latency on /tmp using 10 requests of 1 megabyte each:

```
ioping -c 10 -s 1M /tmp
```

– Measure disk seek rate on /dev/sda:

```
ioping -R /dev/sda
```

– Measure disk sequential speed on /dev/sda:

```
ioping -RL /dev/sda
```

ipcs

Display information about ressources used in IPC (Inter-process Communication).

– Specific information about the Message Queue which has the id 32768:

```
ipcs -qi 32768
```

– General information about all the IPC:

```
ipcs -a
```

jar

Java Applications/Libraries Packager.

– Unzip .jar/.war file to the current directory:

```
jar -xvf *.jar
```

java

Java Application Launcher.

- Execute a java .class file that contains a main method by using just the class name:

```
java filename
```

- Execute a .jar program:

```
java -jar filename.jar
```

- Display JDK, JRE and HotSpot versions:

```
java -version
```

javac

Java Application Compiler.

- Compile a .java file:

```
javac filename.java
```

- Compile several .java files:

```
javac filename1.java filename2.java filename3.java
```

- Compile all .java files in current directory:

```
javac *.java
```

- Compile a .java file and place the resulting class file in a specific directory:

```
javac -d path/to/some/directory filename.java
```

jhat

Java Heap Analysis Tool.

- Analyze a heap dump (from jmap), view via http on port 7000:

```
jhat dump_file.bin
```

- Analyze a heap dump, specifying an alternate port for the http server:

```
jhat -p port dump_file.bin
```

- Analyze a dump letting jhat use up to 8GB RAM (2-4x dump size recommended):

```
jhat -J-mx8G dump_file.bin
```

jmap

Java Memory Map Tool.

- Print shared object mappings for a java process (output like pmap):

```
jmap java_pid
```

- Print heap summary information:

```
jmap -heap filename.jar java_pid
```

- Print histogram of heap usage by type:

```
jmap -histo java_pid
```

- Dump contents of the heap into a binary file for analysis with jhat:

```
jmap -dump:format=b,file=filename java_pid
```

jobs

Display status of jobs in the current session.

- Show status of all jobs:

```
jobs
```

- Show status of a particular job:

```
jobs job_id
```

- Show status and process IDs of all jobs:

```
jobs -l
```

- Show process IDs of all jobs:

```
jobs -p
```

jstack

Java Stack Trace Tool.

- Print java stack traces for all threads in a java process:

```
jstack java_pid
```

- Print mixed mode (java/c++) stack traces for all threads in a java process:

```
jstack -m java_pid
```

- Print stack traces from java core dump:

```
jstack /usr/bin/java file.core
```

kill

Sends a signal to a process. Mostly used for stopping processes.

- Kill the process:

```
kill process_id
```

- List signal names:

```
kill -l
```

last

View the last logged in users.

- View last logins, their duration and other information as read from /var/log/wtmp:

```
last
```

- Specify how many of the last logins to show:

```
last -n login_count
```

- View full login times and dates:

```
last -F
```

- View the last login by a specific user:

```
last user_name
```

- View the last reboot (last login of the pseudo user reboot):

```
last reboot
```

- View the last shutdown (last login of the pseudo user shutdown):

```
last shutdown
```

latexmk

Compile LaTeX source files into finished documents. Automatically does multiple runs when needed.

- Compile a dvi (DeVice Independent file) document from every source:

  `latexmk`

- Compile a dvi document from a specific source file:

  `latexmk source.tex`

- Compile a pdf document:

  `latexmk -pdf source.tex`

- Clean up all temporary tex files in the folder:

  `latexmk -c`

- Clean up temporary tex files created for a specific tex file:

  `latexmk -c source.tex`

- Clean up temporary tex and output files:

  `latexmk -C`

less

Opens a file for reading. Allows movement and search. Doesn't read the entire file (suitable for logs).

- Open a file:

  `less source_file`

- Page up / down:

  `d (next), D (previous)`

- Go to start / end of file:

  `g (start), G (end)`

- Forward search for a string:

  `/something   then   n (next), N (previous)`

- Backward search for a string:

  `?something   then   n (next), N (previous)`

- Exit:

  `q`

license

Create license files for open-source projects.

- Create a license:

```
license license_name
```

- Create a license with custom filename:

```
license -o filename.txt license_name
```

- List all locally available licenses:

```
license ls
```

- Create a license with explicitly-set name and year:

```
license --name author --year release_year license_name
```

ln

Creates links to files and folders.

- Create a symbolic link to a file (or folder):

```
ln -s path/to/file path/to/symlink
```

- Overwrite an existing symbolic to point to a different file:

```
ln -sf path/to/new/file path/to/symlink
```

- Create a hard link to a file:

```
ln path/to/file path/to/hardlink
```

lp

Print files.

- Print the output of a command to the default printer (see `lpstat` command):

  ```
  echo "test" | lp
  ```

- Print a file to the default printer:

  ```
  lp path/to/filename
  ```

- Print a file to a named printer (see `lpstat` command):

  ```
  lp -d printer_name path/to/filename
  ```

- Print N copies of file to default printer (replace N with desired number of copies):

  ```
  lp -n N path/to/filename
  ```

- Print only certain pages to the default printer (print pages 1, 3-5, and 16):

  ```
  lp -P 1,3-5,16 path/to/filename
  ```

lpstat

Show status information about printers.

- List printers present on the machine and whether they are enabled for printing:

  ```
  lpstat -p
  ```

- Show the default printer:

  ```
  lpstat -d
  ```

- Display all available status information:

  ```
  lpstat -t
  ```

- Show a list of print jobs queued by the specified user:

  ```
  lpstat -u user
  ```

ls

List directory contents.

- List files one per line:

```
ls -1
```

- List all files, including hidden files:

```
ls -a
```

- Long format list (permissions, ownership, size and modification date) of all files:

```
ls -la
```

- Long format list with size displayed using human readable units (KB, MB, GB):

```
ls -lh
```

- Long format list sorted by size (descending):

```
ls -lS
```

- Long format list of all files, sorted by modification date (oldest first):

```
ls -ltr
```

lsof

Lists open files and the corresponding processes. Note: In most cases, you need root privilege (or use sudo) because you want to list files opened by others.

- Find the processes that have a given file open:

```
lsof /path/to/file
```

- Find the process that opened a local internet port:

```
lsof -i :port
```

- Only output the process PID:

```
lsof -t /path/to/file
```

- List files opened by the given user:

```
lsof -u username
```

- List files opened by the given command or process:

```
lsof -c process_or_command_name
```

lwp-request

Simple command-line HTTP client. Built with libwww-perl.

- Make a simple GET request:

  `lwp-request -m GET http://example.com/some/path`

- Upload a file with a POST request:

  `cat /path/to/file | lwp-request -m POST http://example.com/some/path`

- Make a request with a custom user agent:

  `lwp-request -H 'User-Agent: user_agent -m METHOD http://example.com/some/path`

- Make a request with HTTP authentication:

  `lwp-request -C username:password -m METHOD http://example.com/some/path`

- Make a request and print request headers:

  `lwp-request -U -m METHOD http://example.com/some/path`

- Make a request and print response headers and status chain:

  `lwp-request -E -m METHOD http://example.com/some/path`

mailx

Send and receive mail.

- To send mail, the content is typed after the command and ended with Control-D:

  `mailx -s "subject" to_addr`

- Send mail with short content:

  `echo "content" | mailx -s "subject" to_addr`

- Send mail with content which written in a file:

  `mailx -s "subject" to_addr < content.txt`

- Send mail to a recipient and CC to another address:

  `mailx -s "subject" -c cc_addr to_addr`

- Send mail and set sender address:

  `mailx -s "subject" -r from_addr to_addr`

- Send mail with an attachment:

  `mailx -a file -s "subject" to_addr`

make

Task runner for rules described in Makefile. Mostly used to control the compilation of an executable from source code.

- Call the all rule:

```
make
```

- Call a specific rule:

```
make rule
```

- Use specific Makefile:

```
make -f file
```

- Execute make from another directory:

```
make -C directory
```

man

Format and display manual pages.

- Display man page for a command:

```
man command
```

- Display path searched for manpages:

```
man --path
```

- Display location of a manpage rather than the manpage itself:

```
man -w command
```

- Do a keyword search for manpages containing a search string:

```
man -k keyword
```

mitmdump

View, record, and programmatically transform HTTP traffic. The command-line counterpart to mitmproxy.

- Start a proxy and save all output to a file:

```
mitmdump -w filename
```

- Filter a saved traffic file to just POST requests:

```
mitmdump -nr input_filename -w output_filename "~m post"
```

- Replay a saved traffic file:

```
mitmdump -nc filename
```

mitmproxy

An interactive man-in-the-middle HTTP proxy.

- Start mitmproxy with default settings:

```
mitmproxy
```

- Start mitmproxy bound to custom address and port:

```
mitmproxy -b ip_address -p port
```

mkdir

Creates a directory.

- Create a directory in current folder or given path:

```
mkdir directory
```

- Create directories recursively (useful for creating nested dirs):

```
mkdir -p path
```

mkfifo

Makes FIFOs (named pipes).

- Create a named pipe at a given path:

```
mkfifo path/to/pipe
```

mmv

Move and rename files in bulk.

- Rename all files with a certain extension to a different extension:

```
mmv "*.old_extension" "#1.new_extension"
```

- Copy report6part4.txt to ./french/rapport6partie4.txt along with all similarly named files:

```
mmv -c "report*part*.txt" "./french/rapport#1partie#2.txt"
```

- Append all .txt files into one file:

```
mmv -a "*.txt" "all.txt"
```

- Convert dates in filenames from "M-D-Y" format to "D-M-Y" format:

```
mmv "[0-1][0-9]-[0-3][0-9]-[0-9][0-9][0-9][0-9].txt" "#3#4-#1#2-#5#6#7#8.txt
"
```

mocha

Execute Mocha JavaScript test runner.

- Run tests with default configuration or as configured in `mocha.opts`:

```
mocha
```

- Run tests contained at a specific location:

```
mocha folder/with/tests
```

- Run tests that match a specific grep pattern:

```
mocha --grep ^regex$
```

- Run tests on changes to JavaScript files in the current directory and once initially:

```
mocha --watch
```

- Run tests with a specific reporter:

```
mocha --reporter reporter
```

montage

Imagemagick image montage tool. Tiles images into a customisable grid.

- Tile images into a grid, automatically resizing images larger than the grid cell size:

```
montage image1.png image2.jpg imageN.png montage.jpg
```

- Tile images into a grid, automatically calculating the grid cell size from the largest image:

```
montage image1.png image2.jpg imageN.png -geometry +0+0 montage.jpg
```

- Set the grid cell size and resize images to fit it before tiling:

```
montage image1.png image2.jpg imageN.png -geometry 640x480+0+0 montage.jpg
```

- Limit the number of rows and columns in the grid, causing input images to overflow into multiple output montages:

```
montage image1.png image2.jpg imageN.png -geometry +0+0 -tile 2x3 montage_%d
.jpg
```

- Resize and crop images to completely fill their grid cells before tiling:

```
montage image1.png image2.jpg imageN.png -geometry +0+0 -resize 640x480^ -gravity
 center -crop 640x480+0+0 montage.jpg
```

more

Opens a file for reading. Allows movement and search in forward direction only. Doesn't read the entire file (suitable for logs).

- Open a file:

```
more source_file
```

- Page down:

```
d (next)
```

- Search for a string:

```
/something   then   n (next)
```

- Exit:

```
q
```

mount

Provides access to an entire filesystem in one directory.

- Show all mounted filesystems:

```
mount
```

- Mount a device:

```
mount -t filesystem_type path_to_device_file directory_to_mount_to
```

- Mount a CD-ROM device (with the filetype ISO9660) to /cdrom (readonly):

```
mount -t iso9660 -o ro /dev/cdrom /cdrom
```

- Mount all the filesystem defined in /etc/fstab:

```
mount -a
```

- Mount a specific filesystem described in /etc/fstab (e.g. "/dev/sda1 /my_drive ext2 defaults 0 2"):

```
mount /my_drive
```

MP4Box

MPEG-4 Systems Toolbox - Muxes streams into MP4 container.

- Display information about an existing MP4 file:

```
MP4Box -info filename
```

- Add an SRT subtitle file into an MP4 file:

```
MP4Box -add input-subs.srt:lang=eng -add input.mp4 output.mp4
```

- Combine audio from one file and video from another:

```
MP4Box -add input1.mp4#audio -add input2.mp4#video output.mp4}
```

mtr

Combined traceroute and ping tool.

- Traceroute to a host and continuously ping all intermediary hops:

```
mtr host
```

- Disable IP address and host name mapping:

```
mtr -n host
```

- Generate output after pinging each hop 10 times:

```
mtr -w host
```

- Force IP IPv4 or IPV6:

```
mtr -4 host
```

mv

Move or rename files and directories.

- Move files in arbitrary locations:

```
mv source target
```

- Do not prompt for confirmation before overwriting existing files:

```
mv -f source target
```

- Do not prompt for confirmation before overwriting existing files but write to standard error before overriding:

```
mv -fi source target
```

- Move files in verbose mode, showing files after they are moved:

```
mv -v source target
```

mysql

The MySQL command-line tool.

- Connect to a database:

```
mysql database_name
```

- Connect to a database, user will be prompted for a password:

```
mysql -u user --password database_name
```

- Connect to a database on another host:

```
mysql -h database_host database_name
```

- Execute SQL statements in a script file (batch file):

```
mysql database_name < script.sql
```

mysqldump

Backups mysql databases.

- Create a backup, user will be prompted for a password:

```
mysqldump -u user --password database_name > filename.sql
```

- Restore a backup, user will be prompted for a password:

```
mysql -u user --password database_name < filename.sql
```

nano

Simple, easy to use editor. An enhanced, free Pico clone.

- Start nano in terminal with {filename}:

```
nano filename
```

- Enable smooth scrolling:

```
nano -S filename
```

- Indent new lines to the previous lines' indentation:

```
nano -i filename
```

nc

Reads and writes tcp or udp data.

- Listen on a specified port:

```
nc  -l port
```

- Connect to a certain port (you can then write to this port):

```
nc ip_address port
```

- Set a timeout:

```
nc -w timeout_in_seconds ipaddress port
```

- Serve a file:

```
cat somefile.txt | nc -l port
```

- Receive a file:

```
nc ip_address port > somefile.txt
```

- Server stay up after client detach:

```
nc -k -l port
```

- Client stay up after EOF:

```
nc -q timeout ip_address
```

nginx

Nginx web server.

- Start server with default config:

```
nginx
```

- Start server with custom config file:

```
nginx -c config_file
```

- Start server with a prefix for all relative paths in config file:

```
nginx -c config_file -p prefix/for/relative/paths
```

- Test configuration without affecting the running server:

```
nginx -t
```

- Reload configuration by sending a signal with no downtime:

```
nginx -s reload
```

nice

Execute a program with a custom scheduling priority (niceness). Niceness values range from -20 (the highest priority) to 19 (the lowest).

- Launch a program with altered priority:

```
nice -n niceness_value command
```

nix-env

Manipulate or query Nix user environments.

- Show available package with name or without name:

```
nix-env -qa pkg_name
```

- Show the status of available packages:

```
nix-env -qas
```

- Install package:

```
nix-env -i pkg_name
```

- Uninstall package:

```
nix-env -e pkg_name
```

- Upgrade one package:

```
nix-env -u pkg_name
```

- Upgrade all packages:

```
nix-env -u
```

nmap

Network exploration tool and security / port scanner. Some features only activate when Nmap is run with privileges.

- Try to determine whether the specified hosts are up and what are their names:

```
nmap -sn ip_or_hostname optional_another_address
```

- Like above, but also run a default 1000-port TCP scan if host seems up:

```
nmap ip_or_hostname optional_another_address
```

- Also enable scripts, service detection, OS fingerprinting and traceroute:

```
nmap -A address_or_addresses
```

- Assume good network connection and speed up execution:

```
nmap -T4 address_or_addresses
```

- Scan a specific list of ports (use -p- for all ports 1-65535):

```
nmap -p port1,port2,...,portN address_or_addresses
```

- Perform TCP and UDP scanning (use -sU for UDP only, -sZ for SCTP, -sO for IP):

```
nmap -sSU address_or_addresses
```

node

Server-side JavaScript platform (Node.js).

- Run a JavaScript file:

```
node file.js
```

- Start a REPL (interactive shell):

```
node
```

- Evaluate JavaScript by passing it in the command:

```
node -e "code"
```

nohup

Allows for a process to live when the terminal gets killed.

- Run process that can live beyond the terminal:

```
nohup command options
```

npm

JavaScript and Node.js package manager. Manage Node.js projects and their module dependencies.

- Download and install a module globally:

```
npm install -g module_name
```

- Download all dependencies referenced in package.json:

```
npm install
```

- Download a given dependency, and add it to the package.json:

```
npm install module_name@version --save
```

- Uninstall a module:

```
npm uninstall module_name
```

- List a tree of installed modules:

```
npm list
```

- Interactively create a package.json file:

```
npm init
```

nvm

Node.js version manager. Switch between NodeJS versions: system, node, 0.10, 0.12, 4.2 etc.

- Install a specific version of NodeJS:

```
nvm install node_version
```

- Use a specific version NodeJS in the current shell:

```
nvm use node_version
```

- Set the default NodeJS version:

```
nvm alias default node_version
```

- List all available NodeJS versions and print the default one:

```
nvm list
```

- Run a specific version NodeJS REPL:

```
nvm run node_version --version
```

- Run app in a specific version of NodeJS:

```
nvm exec node_version node app.js
```

openssl

OpenSSL cryptographic toolkit.

- Generate a 2048bit RSA private key and save it to a file:

```
openssl genrsa -out filename.key 2048
```

- Generate a certificate signing request to be sent to a certificate authority:

```
openssl req -new -sha256 -key filename.key -out filename.csr
```

- Generate a self-signed certificate from a certificate signing request valid for some number of days:

```
openssl x509 -req -days days -in filename.csr -signkey filename.key -out filename
.crt
```

- Display the certificate presented by an SSL/TLS server:

```
openssl s_client -connect host:port </dev/null
```

- Display the complete certificate chain of an HTTPS server:

```
openssl s_client -connect host:443 -showcerts </dev/null
```

optipng

PNG image file optimization utility.

- Compress a PNG with default settings:

  ```
  optipng file.png
  ```

- Compress a PNG with best compression:

  ```
  optipng -o7 file.png
  ```

- Compress a PNG with fastest compression:

  ```
  optipng -o0 file.png
  ```

- Compress a PNG and add interlacing:

  ```
  optipng -i 1 file.png
  ```

- Compress a PNG and remove all metadata:

  ```
  optipng -strip all file.png
  ```

pandoc

General markup converter.

- Convert file to pdf (the output format is automatically determined from the output file's extension):

  ```
  pandoc input.md -o output.pdf
  ```

GNU Parallel

Run commands on multiple CPU cores.

- Gzip several files at once, using all cores:

  ```
  parallel gzip ::: file1 file2 file3
  ```

- Read arguments from stdin, run 4 jobs at once:

  ```
  ls *.txt | parallel -j4 gzip
  ```

- Convert JPG images to PNG using replacement strings:

  ```
  parallel convert {} {.}.png ::: *.jpg
  ```

- Parallel xargs, cram as many args as possible onto one command:

  ```
  args | parallel -X command
  ```

- Break stdin into ~1M blocks, feed each block to stdin of new command:

  ```
  cat bigfile.txt | parallel --pipe --block 1M command
  ```

- Run on multiple machines via SSH:

  ```
  parallel -S machine1,machine2 command ::: arg1 arg2
  ```

pass

Safely store and read passwords or other sensitive data easily. All data is GPG-encrypted, and managed with a git repository.

- Initialize the storage using a gpg-id for encryption:

  ```
  pass init gpg_id
  ```

- Save a new password (prompts you for the value without echoing it):

  ```
  pass insert path/to/data
  ```

- Copy a password (first line of the data file) to the clipboard:

  ```
  pass -c path/to/data
  ```

- List the whole store tree:

  ```
  pass
  ```

- Generate a new random password with a given length, and copy it to the clipboard:

  ```
  pass generate -c path/to/data num
  ```

- Run any git command against the underlying store repository:

  ```
  pass git git-arguments
  ```

passwd

Passwd is a tool used to change a user’s password.

– Change the password of the current user:

```
passwd new password
```

– Change the password of the specified user:

```
passwd username new password
```

– Get the current statuts of the user:

```
passwd -S
```

– Make the password of the account blank (it will set the named account passwordless):

```
passwd -d
```

paste

Merge lines of files.

– Join all the lines into a single line, using TAB as delimiter:

```
paste -s file
```

– Join all the lines into a single line, using the specified delimiter:

```
paste -s -d delimiter file
```

– Merge two files side by side, each in its column, using TAB as delimiter:

```
paste file1 file2
```

– Merge two files side by side, each in its column, using the specified delimiter:

```
paste -d delimiter file1 file2
```

– Merge two files, with lines added alternatively:

```
paste -d '\n' file1 file2
```

patch

Patch a file (or files) with a diff file. Note that diff files contain both the target filenames and list of changes.

- Apply a patch:

```
patch < patchfile.diff
```

- Apply a patch to current directory:

```
patch -p1 < patchfile.diff
```

- Apply the reverse of a patch:

```
patch -R < patchfile.diff
```

pdflatex

Compile a pdf document from LaTeX source files.

- Compile a pdf document:

```
pdflatex source.tex
```

- Compile a pdf document, halting on each error:

```
pdflatex -halt-on-error source.tex
```

pg_dump

Extract a PostgreSQL database into a script file or other archive file.

- Dump database into a SQL-script file:

```
pg_dump db_name > output_file.sql
```

- Same as above, customize username:

```
pg_dump -U username db_name > output_file.sql
```

- Same as above, customize host and port:

```
pg_dump -h host -p port db_name > output_file.sql
```

- Dump a database into a custom-format archive file:

```
pg_dump -Fc db_name > output_file.dump
```

pg_restore

Restore a PostgreSQL database from an archive file created by pg_dump.

- Restore an archive into an existing database:

```
pg_restore -d db_name archive_file.dump
```

- Same as above, customize username:

```
pg_restore -U username -d db_name archive_file.dump
```

- Same as above, customize host and port:

```
pg_restore -h host -p port -d db_name archive_file.dump
```

- Clean database objects before creating them:

```
pg_restore --clean -d db_name archive_file.dump
```

- Use multiple jobs to do the restoring:

```
pg_restore -j 2 -d db_name archive_file.dump
```

pgrep

Find or signal process by name.

- Return PIDs of any running processes with a matching command string:

```
pgrep process_name
```

- Search full command line with parameters instead of just the process name:

```
pgrep -f "process_name parameter"
```

- Search for process run by a specific user:

```
pgrep -u root process_name
```

php

PHP Command Line Interface 'CLI'.

- Parse and execute a file:

```
php file
```

- Check syntax (lint):

```
php -l file
```

- Run PHP interactively:

```
php -a
```

- Run PHP code. Notes: a) Don't use <? ?> tags; b) Escape double quotes with backslash:

```
php -r "code"
```

- Start a PHP built-in web server in the current directory:

```
php -S host:port
```

phpize

Prepare a PHP extension for compiling.

- Prepare the PHP extension in the current directory for compiling:

```
phpize
```

- Delete files previously created by phpize:

```
phpize --clean
```

phpunit

PHPUnit command-line test runner.

- Run tests in the current direcotry. Note: Expects you to have a 'phpunit.xml':

```
phpunit
```

- Run tests in a specific file:

```
phpunit path/to/TestFile.php
```

- Run tests annotated with the given group:

```
phpunit --group name
```

- Run tests and generate a coverage report in HTML:

```
phpunit --coverage-html directory
```

pigz

Multithreaded zlib compression utility.

- Compress a file with default options:

```
pigz filename
```

- Compress a file using the best compression method:

```
pigz -9 filename
```

- Compress a file using no compression and 4 processors:

```
pigz -0 -p4 filename
```

- Decompress a file:

```
pigz -d archive.gz
```

- List the contents of an archive:

```
pigz -l archive.tar.gz
```

ping

Send ICMP ECHO_REQUEST packets to network hosts.

- Ping host:

```
ping host
```

- Ping a host only a specific number of times:

```
ping -c count host
```

- Ping host, specifying the interval in seconds between requests (default is 1 second):

```
ping -i seconds host
```

- Ping host without trying to lookup symbolic names for addresses:

```
ping -n host
```

pip

Python package manager.

- Install a package:

```
pip install package_name
```

- Install a specific version of a package:

```
pip install package_name==package_version
```

- Upgrade a package:

```
pip install -U package_name
```

- Uninstall a package:

```
pip uninstall package_name
```

- Save installed packages to file:

```
pip freeze > requirements.txt
```

- Install packages from file:

```
pip install -r requirements.txt
```

pkill

Signal process by name. Mostly used for stopping processes.

- Kill all processes which match:

```
pkill -9 process_name
```

- Kill all processes which match their full command instead of just the process name:

```
pkill -9 -f "command_name"
```

- Send SIGUSR1 signal to processes which match:

```
pkill -USR1 process_name
```

play

Audio player of SoX - Sound eXchange. Plays any audio from the command line. Audioformats are identified by extension.

- Play the given audio file:

```
play audiofile
```

- Play the given audio files:

```
play audiofile1 audiofile2
```

- Play the given audio at twice the speed:

```
play audiofile speed 2.0
```

- Play the given audio in reverse:

```
play audiofile reverse
```

pngcrush

PNG image compression utility.

- Compress a PNG file:

```
pngcrush in.png out.png
```

- Compress all PNGs and output to directory:

```
pngcrush -d path/to/output *.png
```

- Compress PNG file with all 114 available algorithms and pick the best result:

```
pngcrush -rem allb -brute -reduce in.png out.png
```

printf

Format and print text.

- Print a text message:

```
printf "%s\n" "Hello world"
```

- Print an integer in bold blue:

```
printf "\e[1;34m%.3d\e[0m\n" 42
```

- Print a float number with the unicode Euro sign:

```
printf "\u20AC %.2f\n" 123.4
```

- Print a text message composed with environment variables:

```
printf "var1: %s\tvar2: %s\n" "$VAR1" "$VAR2"
```

- Store a formatted message in a variable (does not work on zsh):

```
printf -v myvar "This is %s = %d\n" "a year" 2016
```

ps

Information about running processes.

- List all running processes:

```
ps aux
```

- List all running processes including the full command string:

```
ps auxww
```

- Search for a process that matches a string:

```
ps aux | grep string
```

psql

PostgreSQL command-line client.

- Connect to *database*. It connects to localhost using default port *5432* with default user:

```
psql database
```

- Connect to *database* on given server *host* running on given *port* with *username* given, no password prompt:

```
psql -h host -p port -U username database
```

- Connect to *database*, user will be prompted for password:

```
psql -h host -p port -U username -W database
```

- Run single *query* against the given *database*. Note: useful in shell scripts:

```
psql -c 'query' database
```

- Run several queries against the given *database*. Note: useful in shell scripts:

```
echo 'query1; query2' | psql database
```

pushd

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

- 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
```

pv

Monitor the progress of data through a pipe.

- Print the contents of the file and display a progress bar:

```
pv file
```

- Measure the speed and amount of data flow between pipes (-s is optional):

```
command1 | pv -s expected_amount_of_data_for_eta | command2
```

- Filter a file, see both progress and amount of output data:

```
pv -cN in big_text_file | grep pattern | pv -cN out > filtered_file
```

- Attach to an already running process and see its file reading progress:

```
pv -d PID
```

- Read an erroneous file, skip errors as dd conv=sync,noerror would:

```
pv -EE path_to_faulty_media > image.img
```

- Stop reading after reading specified amount of data, rate limit to 1K/s:

```
pv -L 1K -S maximum_file_size_to_be_read
```

pwd

Print name of current/working directory.

- Print the current directory:

```
pwd
```

- Print the current directory, and resolve all symlinks (i.e. show the "physical" path):

```
pwd -P
```

Python

Python language interpreter.

- Call a Python interactive shell (REPL):

```
python
```

- Execute script in a given Python file:

```
python script.py
```

- Execute Python language single command:

```
python -c command
```

- Run library module as a script (terminates option list):

```
python -m module arguments
```

qemu

Generic machine emulator and virtualizer. Supports a large variety of CPU architectures.

- Create disk image with a specific size (in gigabytes):

```
qemu-img create image_name.img gigabitesG
```

- Boot from image emulating i386 architecture:

```
qemu-system-i386 -hda image_name.img
```

- Boot from image emulating x64 architecture:

```
qemu-system-x86_64 -hda image_name.img
```

- Boot QEMU instance with a live ISO image:

```
qemu-system-i386 -hda image_name.img -cdrom os-image.iso -boot d
```

- Specify amount of RAM for instance:

```
qemu-system-i386 -m 256 -hda image_name.img -cdrom os-image.iso -boot d
```

- Boot from physical device (e.g. from USB to test bootable medium):

```
qemu-system-i386 -hda /dev/storage_device
```

read

BASH builtin for retrieving data from standard input.

- Store data that you type from the keyboard:

```
read variable
```

- Store each of the next lines you enter as values of an array:

```
read -a array
```

- Enable backspace and GNU readline hotkeys when entering input with read:

```
read -e variable
```

- Specify the number of maximum characters to be read:

```
read -n character_count variable
```

- Use a specific character as a delimiter instead of a new line:

```
read -d new_delimiter variable
```

readlink

Follow symlinks and get symlink information.

- Get the actual file to which the symlink points:

```
readlink filename
```

- Get the absolute path to a file:

```
readlink -f filename
```

redis-cli

Opens a connection to a Redis server.

- Connect to the local server:

```
redis-cli
```

- Connect to a remote server on the default port (6379):

```
redis-cli -h host
```

- Connect to a remote server specifying a port number:

```
redis-cli -h host -p port
```

- Specify a password:

```
redis-cli -a password
```

- Execute Redis command:

```
redis-cli redis command
```

redshift

Adjust the color temperature of your screen according to your surroundings.

- Turn on Redshift with 5700K temperature during day and 3600K at night:

```
redshift -t 5700:3600
```

- Turn on Redshift with a manually-specified custom location:

```
redshift -l latitude:longitude
```

- Turn on Redshift with 70% screen brightness during day and 40% brightness at night:

```
redshift -b 0.7:0.4
```

- Turn on Redshift with custom gamma levels (between 0 and 1):

```
redshift -g red:green:blue
```

- Turn on Redshift with a constant unchanging color temperature:

```
redshift -O temperature
```

rename

Renames multiple files.

- Rename files using a Perl Common Regular Expression (substitute 'foo' with 'bar' wherever found):

```
rename 's/foo/bar/' \*
```

- Dry-run - display which renames would occur without performing them:

```
rename -n 's/foo/bar/' \*
```

- Convert filenames to lower case:

```
rename 'y/A-Z/a-z/' \*
```

- Replace whitespace with underscores:

```
rename 's/\s+/_/g' \*
```

renice

Alters the scheduling priority/nicenesses of one or more running processes. Niceness values range from -20 (most favorable to the process) to 19 (least favorable to the process).

- Change priority of a running process:

```
renice -n niceness_value -p pid
```

- Change priority of all processes owned by a user:

```
renice -n niceness_value -u user
```

- Change priority of all processes that belongs to a group:

```
renice -n niceness_value -g group
```

rm

Remove files or directories.

- Remove files from arbitrary locations:

  `rm /path/to/file /otherpath/to/file2`

- Remove recursively a directory and all its subdirectories:

  `rm -r /path/to/folder`

- Remove directory without prompt:

  `rm -rf /path/to/folder`

- Prompt before every removal:

  `rm -i \*`

rmdir

Removes a directory.

- Remove directory, provided it is empty. Use `rm` to remove not empty directories:

  `rmdir directory`

- Remove directories recursively (useful for nested dirs):

  `rmdir -p path`

route

Use route cmd to set the route table .

- Display the information of route table:

  `route -n`

- Add route rule:

  `sudo route add -net ip_address netmask netmask_address gw gw_address`

- Delete route rule:

  `sudo route del -net ip_address netmask netmask_address dev gw_address`

rsync

Transfers a file either to or from a remote host. Does not allow transfer between two remote hosts. Can transfer single files or files matched by pattern.

- Transfer file from local to remote host:

```
rsync path/to/file remote_host_name:remote_host_location
```

- Transfer file from remote host to local:

```
rsync remote_host_name:remote_file_location local_file_location
```

- Transfer file in archive (to preserve attributes) and compressed (zipped) mode:

```
rsync -az path/to/file remote_host_name:remote_host_location
```

- Transfer a directory and all its children from a remote to local:

```
rsync -r remote_host_name:remote_folder_location local_folder_location
```

- Transfer only updated files from remote host:

```
rsync -ru remote_host_name:remote_folder_location local_folder_location
```

- Transfer file over SSH and show progress:

```
rsync -e ssh --progress remote_host_name:remote_file local_file
```

sails

Sails.js is a realtime enterprise level MVC framework built on top of Node.js.

- Start Sails:

```
sails lift
```

- Create new Sails project:

```
sails new projectName
```

- Generate Sails API:

```
sails generate name
```

- Generate Sails Controller:

```
sails generate controller name
```

- Generate Sails Model:

```
sails generate model name
```

salt-key

Invoke salt locally on a salt minion.

- Perform a highstate on this minion:

  ```
  salt-call state.highstate
  ```

- Perform a highstate dry-run, compute all changes but don't actually perform them:

  ```
  salt-call state.highstate test=true
  ```

- Perform a highstate with verbose debugging output:

  ```
  salt-call -l debug state.highstate
  ```

- List this minion's grains:

  ```
  salt-call grains.items
  ```

salt-key

Manages salt minion keys on the salt master. Needs to be run on the salt master, likely as root or with sudo.

- List all accepted, unaccepted and rejected minion keys:

  ```
  salt-key -L
  ```

- Accept a minion key by name:

  ```
  salt-key -a MINION_ID
  ```

- Reject a minion key by name:

  ```
  salt-key -r MINION_ID
  ```

- Print fingerprints of all public keys:

  ```
  salt-key -F
  ```

salt-run

Frontend for executing salt-runners on minions.

- Show status of all minions:

```
salt-run manage.status
```

- Show all minions which are disconnected:

```
salt-run manage.up
```

salt

Execute commands and assert state on remote salt minions.

- List connected minions:

```
salt '*' test.ping
```

- Execute a highstate on all connected minions:

```
salt '*' state.highstate
```

- Upgrade packages using the OS package manager (apt, yum, brew) on a subset of minions:

```
salt '*.domain.com' pkg.upgrade
```

- Execute an arbitrary command on a particular minion:

```
salt 'minion_id' cmd.run "ls "
```

Sass

Converts SCSS or Sass files to CSS.

- Output converted file to stdout:

```
sass inputfile.(scss|sass)
```

- Immediately convert SCSS or Sass file to CSS to specified output file:

```
sass inputfile.(scss|sass) outputfile.css
```

- Watch SCSS or Sass file for changes and output or update CSS file with same filename:

```
sass --watch inputfile.(scss|sass)
```

- Watch SCSS or Sass file for changes and output or update CSS file with specified filename:

```
sass --watch inputfile.(scss|sass):outputfile.css
```

scp

Secure copy. Copy files between hosts using Secure Copy Protocol over SSH.

– Copy a local file to a remote host:

```
scp local_file remote_host:/path/remote_file
```

– Copy a file from a remote host to a local folder:

```
scp remote_host:/path/remote_file /path/local_dir
```

– Recursively copy the contents of a directory on a remote host to a local directory:

```
scp -r /path/local_dir remote_host:/path/remote_dir
```

– Copy a file between two remote hosts transferring through the local host:

```
scp -3 host1:/path/remote_file.ext host2:/path/remote_dir
```

– Use a specific username when connecting to the remote host:

```
scp local_file remote_username@remote_host:/remote/path
```

– Use a specific ssh private key for authentication with the remote host:

```
scp -i ~/.ssh/id_rsa local_file remote_host:/path/remote_file
```

screen

Hold a session open on a remote server. Manage multiple windows with a single SSH connection.

– Start a new screen session:

```
screen
```

– Start a new named screen session:

```
screen -S name
```

– Show open screen sessions:

```
screen -ls
```

– Reattach to an open screen:

```
screen -r screen id
```

– Detach from inside a screen:

```
ctrl+A D
```

– Kill a detached screen:

```
screen -X -S screen id quit
```

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 -r '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/'
  ```

- 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
  ```

seq

Output a sequence of numbers to stdout.

- Sequence from 1 to 10:

  ```
  seq 10
  ```

- Every 3rd number from 5 to 20:

  ```
  seq 5 3 20
  ```

- Separate the output with a space instead of a newline:

  ```
  seq -s " " 5 3 20
  ```

sftp

Secure File Transfer Program. Interactive program to copy files between hosts over SSH. For non-interactive file transfers, see `scp` or `rsync`.

- Connect to a remote server and enter an interactive command mode:

  ```
  sftp remote_user@remote_host
  ```

- Connect using an alternate port:

  ```
  sftp -P remote_port remote_user@remote_host
  ```

- Transfer remote file to the local system:

  ```
  get /path/remote_file
  ```

- Transfer local file to the remote system:

  ```
  put /path/local_file
  ```

- Transfer remote folder to the local system recursively (works with `put` too):

  ```
  get -R /path/remote_folder
  ```

- Get list of files on local machine:

  ```
  lls
  ```

- Get list of files on remote machine:

  ```
  ls
  ```

sh

Bourne shell. The standard command language interpreter.

- Start interactive shell:

  ```
  sh
  ```

- Execute a command:

  ```
  sh -c command
  ```

- Run commands from a file:

  ```
  sh file.sh
  ```

- Run commands from STDIN:

  ```
  sh -s
  ```

shred

Overwrite files to securely delete data.

- Overwrite a file:

```
shred file
```

- Overwrite a file, leaving zeroes instead of random data:

```
shred --zero file
```

- Overwrite a file 25 times:

```
shred -n25 file
```

- Overwrite a file and remove it:

```
shred --remove file
```

skicka

Manage your Google Drive.

- Upload a file/folder to Google Drive:

```
skicka upload path/to/local path/to/remote
```

- Download a file/folder from Google Drive:

```
skicka download path/to/remote path/to/local
```

- List files:

```
skicka ls path/to/folder
```

- Show amount of space used by children folders:

```
skicka du path/to/parent/folder
```

- Create a folder:

```
skicka mkdir path/to/folder
```

- Delete a file:

```
skicka rm path/to/file
```

sl

Steam locomotive running through your terminal.

- Let a steam locomotive run through your terminal:

```
sl
```

- The train burns, people scream:

```
sl -a
```

- Let the train fly:

```
sl -F
```

slackcat

Utility for passing files and command output to Slack.

- Post a file to Slack:

```
slackcat --channel channel_name path/to/file
```

- Post a file to Slack with a custom filename:

```
slackcat --channel channel_name --filename=filename path/to/file
```

- Pipe command output to Slack as a text snippet:

```
command | slackcat --channel channel_name --filename=snippet_name
```

- Stream command output to Slack continuously:

```
command | slackcat --channel channel_name --stream
```

sleep

Delay for a specified amount of time.

- Delay in seconds:

```
sleep seconds
```

- Delay in minutes:

```
sleep minutesm
```

- Delay in hours:

```
sleep hoursh
```

socat

Multipurpose relay (SOcket CAT).

- Listen to a port, wait for an incoming connection and transfer data to STDIO:

```
socat - TCP-LISTEN:8080,fork
```

- Create a connection to a host and port, transfer data in STDIO to connected host:

```
socat - TCP4:www.domain.com:80
```

- Forward incoming data of a local port to another host and port:

```
socat TCP-LISTEN:80,fork TCP4:www.domain.com:80
```

sort

Sort lines of text files.

- Sort a file in ascending order:

```
sort filename
```

- Sort a file in descending order:

```
sort -r filename
```

- Sort passwd file by the 3rd field:

```
sort -t: -k 3n /etc/passwd
```

- Sort a file as number in ascending order:

```
sort -n filename
```

- Sort a file preserving only unique lines:

```
sort -u filename
```

sox

SoX - Sound eXchange. Play, record and convert audio files. Audioformats are identified by extension.

– Merge two audio files into one:

```
sox -m input_audiofile1 input_audiofile2 output_audiofile
```

– Trim an audio file to the specified times:

```
sox input_audiofile output_audiofile trim start end
```

– Normalize an audio file (adjust volume to the maximum peak level, without clipping):

```
sox --norm input_audiofile output_audiofile
```

– Reverse and save an audio file:

```
sox input_audiofile output_audiofile reverse
```

– Print statistical data of an audio file:

```
sox input_audiofile -n stat
```

split

Split a file into pieces.

– Split a file, each split having 10 lines (except the last split):

```
split -l 10 filename
```

– Split a file into 5 files. File is split such that each split has same size (except the last split):

```
split -n 5 filename
```

– Split a file with at most 512 bytes of lines in each split:

```
split -C 512 filename
```

srm

Securely remove files or directories. Overwrites the existing data one or multiple times. Drop in replacement for rm.

- Remove a file after a single-pass overwriting with random data:

  ```
  srm -s /path/to/file
  ```

- Remove a file after seven passes of overwriting with random data:

  ```
  srm -m /path/to/file
  ```

- Recursively remove a directory and its contents overwriting each file with a single-pass of random data:

  ```
  srm -r -s /path/to/folder
  ```

- Prompt before every removal:

  ```
  srm -i \*
  ```

ssh-keygen

Generate ssh keys user for authentication, password-less logins, and other things.

- Generate a key interactively:

  ```
  ssh-keygen
  ```

- Specify file in which to save the key:

  ```
  ssh-keygen -f ~/.ssh/filename
  ```

- Generate a DSA 2048 bit (default) key:

  ```
  ssh-keygen -t dsa
  ```

- Generate an RSA 4096 bit key with your email as a comment:

  ```
  ssh-keygen -t rsa -b 4096 -C "email"
  ```

- Retrieve the key fingerprint from a host (useful for confirming the authenticity of the host when first connecting to it via SSH):

  ```
  ssh-keygen -l -F remote_host
  ```

SSH

Secure Shell is a protocol used to securely log onto remote systems. It can be used for logging or executing commands on a remote server.

- Connect to a remote server:

```
ssh username@remote_host
```

- Connect to a remote server with a specific identity (private key):

```
ssh -i /path/to/key_file username@remote_host
```

- Connect to a remote server using a specific port:

```
ssh username@remote_host -p 2222
```

- Run a command on a remote server:

```
ssh remote_host command -with -flags
```

- SSH tunneling: Dynamic port forwarding (SOCKS proxy on localhost:9999):

```
ssh -D 9999 -C username@remote_host
```

- SSH tunneling: Forward a specific port (localhost:9999 to slashdot.org:80):

```
ssh -L 9999:slashdot.org:80 username@remote_host
```

- SSH enable agent forward:

```
ssh -A username@remote_host
```

SSHFS

Filesystem client based on ssh.

- Mount remote directory:

```
sshfs username@remote_host:remote_directory mountpoint
```

- Unmount remote directory:

```
fusermount -u mountpoint
```

- Mount remote directory from server with specific port:

```
sshfs username@remote_host:remote_directory -p 2222
```

- Use compression:

```
sshfs username@remote_host:remote_directory -C
```

strings

Find printable strings in an object file or binary.

- Print all strings in a binary:

  ```
  strings file
  ```

- Limit results to strings at least *length* characters long:

  ```
  strings -n length file
  ```

- Prefix each result with its offset within the file:

  ```
  strings -t d file
  ```

- Prefix each result with its offset within the file in hexadecimal:

  ```
  strings -t x file
  ```

su

Switch shell to another user.

- Switch to user {{username}} (password required):

  ```
  su username
  ```

- Switch to superuser (admin password required):

  ```
  su
  ```

- Switch to user {{username}} and simulate a full login shell:

  ```
  su - username
  ```

sudo

Execute a command as another user.

- List of an unreadable directory:

  ```
  sudo ls /usr/local/scrt
  ```

- To edit a file as user www:

  ```
  sudo -u www vi /var/www/index.html
  ```

- To shutdown the machine:

  ```
  sudo shutdown -r +10 "Cya soon!"
  ```

- To repeat the last command as sudo:

  ```
  sudo !!
  ```

sum

Compute checksums and the number of blocks for a file. A predecessor to the more modern `cksum`.

- Compute a checksum with BSD-compatible algorithm and 1024-byte blocks:

```
sum file
```

- Compute a checksum with System V-compatible algorithm and 512-byte blocks:

```
sum --sysv file
```

svn

Subversion command line client tool.

- Check out a working copy from a repository:

```
svn co url/to/repository
```

- Bring changes from the repository into the working copy:

```
svn up
```

- Put files and directories under version control, scheduling them for addition to repository. They will be added in next commit:

```
svn add PATH...
```

- Send changes from your working copy to the repository:

```
svn ci -m commit log message [PATH...]
```

- Show detailed help:

```
svn help
```

tabula

Extract tables from PDF files.

- Extract all tables from a PDF to a CSV file:

```
tabula -o file.csv file.pdf
```

- Extract all tables from a PDF to a JSON file:

```
tabula --format JSON -o file.json file.pdf
```

- Extract tables from pages 1, 2, 3, and 6 of a PDF:

```
tabula --pages 1-3,6 file.pdf
```

- Extract tables from page 1 of a PDF, guessing which portion of the page to examine:

```
tabula --guess --pages 1 file.pdf
```

- Extract all tables from a PDF, using ruling lines to determine cell boundaries:

```
tabula --spreadsheet file.pdf
```

- Extract all tables from a PDF, using blank space to determine cell boundaries:

```
tabula --no-spreadsheet file.pdf
```

tac

Print and concatenate files in reverse.

- Print the contents of *file1* reversed to the standard output:

```
tac file1
```

- Concatenate several files reversed into the target file:

```
tac file1 file2 > target-file
```

tail

Display the last part of a file.

- Show last ‘num’ lines in file:

  ```
  tail -n num file
  ```

- Show all file since line ‘num’:

  ```
  tail -n +num file
  ```

- Show last ‘num’ bytes in file:

  ```
  tail -c num file
  ```

- Keep reading file until ctrl-c:

  ```
  tail -f file
  ```

tar

Archiving utility. Optional compression with gzip / bzip.

- Create an archive from files:

  ```
  tar cf target.tar file1 file2 file3
  ```

- Create a gzipped archive:

  ```
  tar czf target.tar.gz file1 file2 file3
  ```

- Extract an archive in a target folder:

  ```
  tar xf source.tar -C folder
  ```

- Extract a gzipped archive in the current directory:

  ```
  tar xzf source.tar.gz
  ```

- Extract a bzipped archive in the current directory:

  ```
  tar xjf source.tar.bz2
  ```

- Create a compressed archive, using archive suffix to determine the compression program:

  ```
  tar caf target.tar.xz file1 file2 file3
  ```

- List the contents of a tar file:

  ```
  tar tvf source.tar
  ```

tcpdump

Dump traffic on a network.

- Capture the traffic of a specific interface:

```
tcpdump -i eth0
```

- Capture all TCP traffic showing contents (ASCII) in console:

```
tcpdump -A tcp
```

- Capture the traffic from or to a host:

```
tcpdump host www.example.com
```

- Capture the traffic from a specific interface, source, destination and destination port:

```
tcpdump -i eth0 src 192.168.1.1 and dst 192.168.1.2 and dst port 80
```

- Capture the traffic of a network:

```
tcpdump net 192.168.1.0/24
```

- Capture all traffic except traffic over port 22 and save to a dump file:

```
tcpdump -w dumpfile.pcap not port 22
```

tee

Read from standard input and write to standard output and files.

- Copy standard input to each FILE, and also to standard output:

```
echo "example" | tee FILE
```

- Append to the given FILEs, do not overwrite:

```
echo "example" | tee -a FILE
```

telnet

Telnet is used to connect to a specified port of a host.

- Telnet to a certain port:

```
telnet  ip_address port
```

- To exit a telnet session:

```
quit
```

- Default escape character:

```
CTRL + ]
```

- Specify an escape character (x is the escape character):

```
telnet -e x ip_address port
```

test

Evaluate condition. If it is true, returns 0 exit status, otherwise returns 1.

- Test if given variable is equal to given string:

```
test $MY_VAR == '/bin/zsh'
```

- Test if given variable is empty:

```
test -z $GIT_BRANCH
```

- Test if file exists:

```
test -e filename
```

- Test if directory not exists:

```
test ! -d path/to/directory
```

- If-else statement:

```
test condition && echo "true" || echo "false"
```

time

See how long a command takes.

- Time "ls":

```
time ls
```

tldr

Simplified man pages.

- Get typical usages of a command (hint: this is how you got here!):

```
tldr command
```

- Update the local cache of tldr pages:

```
tldr --update
```

tldrl

Lint and format TLDR pages.

- Lint all pages:

```
tldrl pages_directory
```

- Format a specific page to stdout:

```
tldrl -f page.md
```

- Format all pages in place:

```
tldrl -fi pages_directory
```

tmux

Multiplex several virtual consoles.

- Start a new tmux session:

```
tmux
```

- Start a new named tmux session:

```
tmux new -s name
```

- List sessions:

```
tmux ls
```

- Attach to a session:

```
tmux a
```

- Attach to a named session:

```
tmux a -t name
```

- Detach from session:

```
ctrl+b d
```

- Kill session:

```
tmux kill-session -t name
```

touch

Change a file access and modification times (atime, mtime).

- Create a new empty file(s) or change the times for existing file(s) to current time:

```
touch filename
```

- Set the times on a file to a specific date and time:

```
touch -t YYYYMMDDHHMM.SS filename
```

- Use the times from a file to set the times on a second file:

```
touch -r filename filename2
```

tput

View and modify terminal settings and capabilities.

- Move the cursor to a screen location:

```
tput cup y_coordinate x_coordinate
```

- Set foreground (af) or background (ab) color:

```
tput setaf|setab ansi_color_code
```

- Show number of columns, lines, or colors:

```
tput cols|lines|colors
```

- Ring the terminal bell:

```
tput bel
```

- Reset all terminal attributes:

```
tput sgr0
```

tr

Translate characters - run replacements based on single characters and character sets.

- Replace all occurrences of a character in a file, and print the result:

```
tr find_characters replace_characters < filename
```

- Map each character of the first set to the corresponding character of the second set:

```
tr 'abcd' 'jkmn' < filename
```

- Delete all occurances of the specified set of characters from the input:

```
tr -d 'input_characters'
```

- Compress a series of identical characters to a single character:

```
tr -s '\n'
```

- Translate the contents of the file to upper-case and print result:

```
tr "[:lower:]" "[:upper:]" < filename
```

- Strip out non-printable characters from the file and print result:

```
tr -cd "[:print:]" < filename
```

traceroute

Print the route packets trace to network host.

– Traceroute to a host:

```
traceroute host
```

– Disable IP address and host name mapping:

```
traceroute -n host
```

– Specify wait time for response:

```
traceroute -w 0.5 host
```

– Specify number of queries per hop:

```
traceroute -q 5 host
```

transcode

Suite of command line utilities for transcoding video and audio codecs. And converting between formats.

– Create stabilisation file to be able to remove camera shakes:

```
transcode -J stabilize -i inputfile
```

– Remove camera shakes after creating stabilisation file, transform video using xvid:

```
transcode -J transform -i inputfile -y xvid -o outputfile
```

– Resize the video to 640x480 pixels and convert to MPEG4 codec using xvid:

```
transcode -Z 640x480 -i inputfile -y xvid -o outputfile
```

tree

Show the contents of the current directory as a tree.

- Show files and directories with a depth of ‘num’:

  ```
  tree -L num
  ```

- Show directories only:

  ```
  tree -d
  ```

- Show hidden files too:

  ```
  tree -a
  ```

- Print human readable size of files:

  ```
  tree -h
  ```

- Print the full path for each file:

  ```
  tree -f
  ```

- Print the tree without lines and indentation. Useful when used with -f:

  ```
  tree -i
  ```

tty

Returns terminal name.

- Print the file name of this terminal:

  ```
  tty
  ```

ufraw-batch

Convert RAW files from cameras into standard image files.

- Simply convert RAW files to jpg:

  ```
  ufraw-batch --out-type=jpg input-file(s)
  ```

- Simply convert RAW files to png:

  ```
  ufraw-batch --out-type=png input-file(s)
  ```

- Extract the preview image from the raw file:

  ```
  ufraw-batch --embedded-image input-file(s)
  ```

- Save the file with size up to the given maximums MAX1 and MAX2:

  ```
  ufraw-batch --size=MAX1,MAX2 input-file(s)
  ```

umount

Revokes access to an entire filesystem mounted to a directory. A filesystem cannot be unmounted when it is busy.

– Unmount a filesystem:

```
umount path_to_device_file
```

– OR:

```
umount path_to_mounted_directory
```

– Unmount all mounted filesystems (dangerous!):

```
umount -a
```

uname

Print details about the current machine and the operating system running on it. Note: If you're on Linux, try also the `lsb_release` command.

– Print hardware-related information: machine and processor:

```
uname -mp
```

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

```
uname -srv
```

– Print the nodename (hostname) of the system:

```
uname -n
```

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

```
uname -a
```

uniq

Output the unique lines from the given input or file.

- Display each line once:

```
uniq file
```

- Display only unique lines:

```
uniq -u file
```

- Display only duplicate lines:

```
uniq -d file
```

- Display number of occurences of each line along with that line:

```
uniq -c file
```

unrar

Extract RAR archives.

- Extract files with original directory structure:

```
unrar x compressed.rar
```

- Extract files into current directory, losing directory structure in the archive:

```
unrar e compressed.rar
```

- Test integrity of each file inside the archive file:

```
unrar t compressed.rar
```

- List files inside the archive file without decompressing it:

```
unrar l compressed.rar
```

unzip

Extract compressed files in a ZIP archive.

- Extract zip file(s) (for multiple files, seperate file paths by spaces):

  ```
  unzip file(s)
  ```

- Extract zip files(s) to given path:

  ```
  unzip files(s) -d /path/to/put/extracted/files
  ```

- List the contents of a zip file without extracting:

  ```
  unzip -l file
  ```

uptime

Tell how long the system has been running and other information.

- Print current time, uptime, number of logged-in users and other information:

  ```
  uptime
  ```

vagrant

Manage lightweight, reproducible, and portable development environments.

- Create Vagrantfile in current folder with the base Vagrant box:

  ```
  vagrant init
  ```

- Create Vagrantfile with the Ubuntu 14.04 (Trusty Tahr) box from HashiCorp Atlas:

  ```
  vagrant init ubuntu/trusty32
  ```

- Start and provision the vagrant environment:

  ```
  vagrant up
  ```

- Suspend the machine:

  ```
  vagrant suspend
  ```

- Connect to machine via SSH:

  ```
  vagrant ssh
  ```

vim

Vi IMproved, a programmer’s text editor.

- Open a file with cursor at the given line number:

  `vim file +linenumber`

- Open multiple files at once, each file in it’s own tab page:

  `vim -p file1 file2 file3`

- Open a file in read-only mode:

  `view file`

- Exit vim:

  `[Esc] (to switch to normal mode), then :q`

vimtutor

Vim tutor, teaching the basic vim commands.

- Launch the vim tutor using the given language (en, fr, de, …):

  `vimtutor language`

- Exit the tutor:

  `[Esc] (to switch to normal mode), then :q`

virtualenv

Create virtual isolated Python environments.

- Create a new environment:

  `virtualenv path/to/venv`

- Start (select) the environment:

  `source path/to/venv/bin/activate`

- Stop the environment:

  `deactivate`

visudo

Safely edit the sudoers file.

- Edit sudoers file:

```
sudo visudo
```

- Check sudoers file for errors:

```
sudo visudo -c
```

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
```

watch

Execute a program periodically, showing output fullscreen.

- Repeatedly run a command and show the result:

```
watch command
```

- Re-run a command every 60 seconds:

```
watch -n 60 command
```

- Monitor the contents of a directory, highlighting differences as they appear:

```
watch -d ls -l
```

wc

Count words, bytes, or lines.

- Count lines in file:

```
wc -l file
```

- Count words in file:

```
wc -w file
```

- Count characters (bytes) in file:

```
wc -c file
```

- Count characters in file (taking multi-byte character sets into account):

```
wc -m file
```

wget

Download files from the Web. Supports HTTP, HTTPS, and FTP.

- Download a URL to a file:

```
wget -O filename "url"
```

- Limit download speed:

```
wget --limit-rate=200k url
```

- Continue an incomplete download:

```
wget -c url
```

- Download a full website:

```
wget --mirror -p --convert-links -P target_folder url
```

- FTP download with username and password:

```
wget --ftp-user=username --ftp-password=password url
```

which

Locate the a program in the user's path.

- Search the PATH environment variable and display the location of any matching executables:

```
which executable
```

- If there are multiple executables which match, display all:

```
which -a executable
```

while

Simple shell loop.

- Read stdin and perform an action on every line:

```
while read line; do echo "$line"; done
```

- Execute a command forever once every second:

```
while :; do command; sleep 1; done
```

who

Display who is logged in and related data (processes, boot time).

- Display the username, line, and time of all currently logged-in sessions:

```
who
```

- Display information only for the current terminal session:

```
who am i
```

- Display all available information:

```
who -a
```

- Display all available information with table headers:

```
who -a -H
```

whoami

Show the username of the current user.

- Display currently logged user name:

```
whoami
```

x_x

View Excel and CSV files from the command-line.

- View an XLSX or CSV file:

```
x_x file.ext
```

- View an XLSX or CSV file, using the first row as table headers:

```
x_x -h 0 file.ext
```

- View a CSV file with unconventional delimiters:

```
x_x --delimiter=';' --quotechar='|' file.csv
```

xargs

Execute a command with piped arguments.

- Main use:

```
arguments | xargs command
```

- Handle whitespace in arguments:

```
arguments_null_terminated | xargs -0 command
```

- Delete all files that start with ‘M’:

```
find . -name 'M*' | xargs rm
```

- Insert arguments at chosen position:

```
arguments | xargs -I piped_arguments command piped_arguments rest_of_arguments
```

xcv

Cut, copy, and paste in the command-line.

– Cut a file:

```
xcv x input_file
```

– Copy a file:

```
xcv c input_file
```

– Paste a file:

```
xcv v output_file
```

– List files available for pasting:

```
xcv l
```

xz

Compress or decompress .xz and .lzma files.

– Compress a file:

```
xz file
```

– Decompress a file:

```
xz -d file.xz
```

– Decompress a file and write to stdout:

```
xz -dc file.xz
```

– Compress a file, but don’t delete the original:

```
xz -k file
```

– Compress a file using the fastest compression:

```
xz -0 file
```

– Compress a file using the best compression:

```
xz -9 file
```

yes

Output something repeatedly.

- Repeatedly output “message”:

```
yes message
```

- Repeatedly output “y”:

```
yes
```

youtube-dl

Download videos from YouTube and other websites.

- Download a video or playlist:

```
youtube-dl https://www.youtube.com/watch?v=oHg5SJYRHA0
```

- Download the audio from a video and convert it to an MP3:

```
youtube-dl -x --audio-format mp3 url
```

- Download video(s) as MP4 files with custom filenames:

```
youtube-dl --format mp4 --output "%(title) by %(uploader) on %(upload_date)
in %(playlist).%(ext)" url
```

- Download a video and save its description, metadata, annotations, subtitles, and thumbnail:

```
youtube-dl --write-description --write-info-json --write-annotations --write
-sub --write-thumbnail url
```

- From a playlist, download all “Let’s Play” videos that aren’t marked “NSFW” or age-restricted for 7 year-olds:

```
youtube-dl --match-title "let's play" --age-limit 7 --reject-title "nsfw" playlist_url
```

zbarimg

Scan and decode bar codes from image file(s).

- Process an image file:

```
zbarimg image file
```

zcat

Print data from gzip compressed files.

- Print the uncompressed contents of a gzipped file to the standard output:

```
zcat file.txt.gz
```

zdb

ZFS debugger.

- Show detailed configuration of all mounted ZFS zpools:

```
zdb
```

- Show detailed configuration for a specific ZFS pool:

```
zdb -C poolname
```

- Show statistics about number, size and deduplication of blocks:

```
zdb -b poolname
```

zfs

Manage ZFS filesystems.

- List all available zfs filesystems:

```
zfs list
```

- Create a new ZFS filesystem:

```
zfs create pool_name/filesystem_name
```

- Delete a ZFS filesystem:

```
zfs destroy pool_name/filesystem_name
```

- Create a Snapshot of a ZFS filesystem:

```
zfs snapshot pool_name/filesystem_name@snapshot_name
```

- Enable compression on a filesystem:

```
zfs set compression=on pool_name/filesystem_name
```

- Change mountpoint for a filesytem:

```
zfs set mountpoint=/my/mount/path pool_name/filesystem_name
```

zip

Package and compress (archive) files into zip file.

- Package and compress multiple directories and files:

```
zip -r compressed.zip /path/to/dir1 /path/to/dir2 /path/to/file
```

- Add files to an existing zip file:

```
zip compressed.zip path/to/file
```

- Remove unwanted files from an existing zip file:

```
zip -d compressed.zip "foo/*.tmp"
```

- Exclude unwanted files from being added to the compressed archive:

```
zip -r compressed.zip path/to/dir -x \*.git\* \*node_modules\* ...
```

zless

View compressed files.

- Page through a compressed archive with `less`:

```
zless file.txt.gz
```

zpool

Manage ZFS pools.

- Show the configuration and status of all ZFS zpools:

```
zpool status
```

- Check a ZFS pool for errors (verifies the checksum of EVERY block). Very CPU and disk intensive:

```
zpool scrub pool_name
```

- List zpools available for import:

```
zpool import
```

- Import a zpool:

```
zpool import pool_name
```

- Export a zpool (unmount all filesystems):

```
zpool export pool_name
```

- Show the history of all pool operations:

```
zpool histrory pool_name
```

- Create a mirrored pool:

```
zpool create pool_name mirror disk1 disk2 mirror disk3 disk4
```

zsh

Z SHell. bash and sh-compatible command line interpreter.

- Start interactive command line interpreter:

```
zsh
```

- Execute command passed as parameter:

```
zsh -c command
```

- Run commands from file (script):

```
zsh file
```

- Run commands from file and print them as they are executed:

```
zsh -x file
```

2 LINUX

apt-cache

Debian and Ubuntu package query tool.

- Search for a package in your current sources:

```
apt-cache search query
```

- Show information about a package:

```
apt-cache show package
```

- Show whether a package is installed and up to date:

```
apt-cache policy package
```

- Show dependencies for a package:

```
apt-cache depends package
```

- Show packages that depend on a particular package:

```
apt-cache rdepends package
```

apt-get

Debian and Ubuntu package management utility.

- Synchronize list of packages and versions available. This should be run first, before running subsequent apt-get commands:

```
apt-get update
```

- Install a new package:

```
apt-get install package
```

- Remove a package:

```
apt-get remove package
```

- Upgrade installed packages to newest available versions:

```
apt-get upgrade
```

- Upgrade installed packages (like "upgrade"), but remove obsolete packages and install additional packages to meet new dependencies:

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

apt-key

Key management utility for the APT Package Manager on Debian and Ubuntu.

- List trusted keys:

```
apt-key list
```

- Add a key to the trusted keystore:

```
apt-key add public_key_file.asc
```

- Delete a key from the trusted keystore:

```
apt-key del key_id
```

- Add a remote key to the trusted keystore:

```
wget -qO - https://host.tld/filename.key | apt-key add -
```

aptitude

Debian and Ubuntu package management utility.

- Synchronize list of packages and versions available. This should be run first, before running subsequent aptitude commands:

```
aptitude update
```

- Install a new package and its dependencies:

```
aptitude install package
```

- Search for a package:

  ```
  aptitude search package
  ```

- Remove a package and all packages depending on it:

  ```
  aptitude remove package
  ```

- Do an `aptitude remove package` and remove all config files:

  ```
  aptitude purge package
  ```

- Upgrade installed packages to newest available versions:

  ```
  aptitude upgrade
  ```

- Upgrade installed packages (like `aptitude upgrade`) including removing obsolete packages and installing additional packages to meet new package dependencies:

  ```
  aptitude full-upgrade
  ```

archey

Simple tool for stylishly displaying system information.

- Show system information:

  ```
  archey
  ```

bzip2

A block-sorting file compressor.

- Compress file:

  ```
  bzip2 path/to/file_to_compress
  ```

- Decompress file:

  ```
  bzip2 -d path/to/compressed_file.bz2
  ```

- Decompress to console:

  ```
  bzip2 -dc path/to/compressed_file.bz2
  ```

chattr

Change attributes of files or folders.

- Make a file or folder immutable to changes and deletion, even by superuser:

```
chattr +i path
```

- Make a file or folder mutable:

```
chattr -i path
```

- Recursively make an entire folder and contents immutable:

```
chattr -R +i folder
```

chroot

Run command or interactive shell with special root directory.

- Run command as new root directory:

```
chroot /path/to/new/root command
```

- Specify user and group (ID or name) to use:

```
chroot --userspec=user:group
```

compose

An alias to a `run-mailcap`'s action compose. Originally `run-mailcap` is used to mime-type/file.

- Compose action can be used to compose any existing file or new on default mailcap edit tool:

```
compose filename
```

- With `run-mailcap`:

```
run-mailcap --action=compose filename
```

dd

Convert and copy a file.

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

```
dd if=file.iso of=/dev/usb_drive status=progress
```

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

```
dd if=/dev/source_drive of=/dev/dest_drive bs=4M conv=noerror status=progress
```

- 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
```

dnf

Package management utility for RHEL, Feodra, and CentOS (replaces yum).

- Synchronize list of packages and versions available. This should be run first, before running subsequent dnf commands:

```
dnf update
```

- Install a new package:

```
dnf install package
```

- Install a new package and assume yes to all questions:

```
dnf -y install package
```

- Remove a package:

```
dnf remove package
```

- Upgrade installed packages to newest available versions:

```
dnf upgrade
```

dpkg-query

A tool that shows information about installed packages.

- List all installed packages:

```
dpkg-query -l
```

- List installed packages matching a pattern:

```
dpkg-query -l 'pattern'
```

- List all files installed by a package:

```
dpkg-query -L package_name
```

- Show information about a package:

```
dpkg-query -s package_name
```

dpkg

Debian package manager.

- Install a package:

```
dpkg -i /path/to/file
```

- Remove a package:

```
dpkg -r package_name
```

- List installed packages:

```
dpkg -l pattern
```

- List package contents:

```
dpkg -L package_name
```

du

Estimate file space usage.

- Get a sum of the total size of a file/folder in human readable units:

```
du -sh file_or_directory
```

- List file sizes of a directory and any subdirectories in KB:

```
du -k file_or_directory
```

- List file sizes of a directory and any subdirectories in MB:

```
du -m file_or_directory
```

- Get recursively, individual file/folder sizes in human readable form:

```
du -ah directory
```

- List the KB sizes of directories for N levels below the specified directory:

```
du --max-depth=N
```

edit

An alias to a `run-mailcap`'s action edit. Originally `run-mailcap` is used to process/edit mime-type/file.

- Edit action can be used to view any file on default mailcap explorer:

```
edit filename
```

- With `run-mailcap`:

```
run-mailcap --action=edit filename
```

emerge

Gentoo Linux package manager utility.

- Synchronize all packages:

```
emerge --sync
```

- Update all packages, including dependencies:

```
emerge -uDNav @world
```

- Resume a failed updated, skipping the failing package:

```
emerge --resume --skipfirst
```

- Install a new package, with confirmation:

```
emerge -av package-name
```

- Remove a package, with confirmation:

```
emerge -Cav package-name
```

- Remove orphaned packages (that were installed only as dependencies):

```
emerge -avc
```

- Search the package database for a keyword:

```
emerge -S keyword
```

expr

Evaluate expressions and manipulate strings.

- Get string length:

```
expr length string
```

- Evaluate logical or math expression with an operator ('+', '-', '*','&','|', etc.). Special symbols should be escaped:

```
expr first_argument operator second_argument
```

- Get position of the first character in 'string' that matches 'substring':

```
echo $(expr index string substring)
```

- Extract part of the string:

```
echo $(expr substr string position_to_start number_of_characters
```

- Extract part of the string which matches a regular expression:

```
echo $(expr string : '\(regular_expression\)')
```

fc-match

Match available fonts.

- Return a sorted list of best matching fonts:

```
fc-match -s 'Font Name'
```

fc-pattern

Shows information about a font matching a pattern.

- Display default infomation about a font:

```
fc-pattern -d 'Font Name'
```

findmnt

Find your filesystem.

- List all mounted filesystems:

```
findmnt
```

- Search for a device:

```
findmnt /dev/sdb1
```

- Search for a mountpoint:

```
findmnt /
```

- Find filesystems in specific type:

```
findmnt -t ext4
```

- Find filesystems with specific label:

```
findmnt LABEL=BigStorage
```

firewall-cmd

The firewalld command line client.

- View the available firewall zones:

  ```
  firewall-cmd --get-active-zones
  ```

- View the rules which are currently applied:

  ```
  firewall-cmd --list-all
  ```

- Permanently open the port for a service in the specified zone (like port 443 when in the public zone):

  ```
  firewall-cmd --permanent --zone=public --add-service=https
  ```

- Permanently close the port for a service in the specified zone (like port 80 when in the public zone):

  ```
  firewall-cmd --permanent --zone=public --remove-service=http
  ```

- Reload firewalld to force rule changes to take effect:

  ```
  firewall-cmd --reload
  ```

free

Display amount of free and used memory in the system.

- Display system memory:

  ```
  free
  ```

- Display memory in Bytes/KB/MB/GB:

  ```
  free -b/-k/-m/-g
  ```

- Display memory in human readable units:

  ```
  free -h
  ```

- Continuous monitor memory (refresh every X seconds):

  ```
  free -s X
  ```

fuser

Display process IDs currently using files or sockets. Require admin privileges.

– Identify process using a TCP socket:

```
fuser -n tcp port
```

getent

Get entries from Name Service Switch libraries.

– Get list of all groups:

```
getent group
```

– See the members of a group:

```
getenet group group_name
```

– Get list of all services:

```
getent services
```

– Find a username by UID:

```
getent passwd 1000
```

– Perform a reverse DNS lookup:

```
getent hosts host
```

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 size_in_bytes filename
```

– Output everything but the last few lines of a file:

```
head -n -count_of_lines filename
```

– Output everything but the last few bytes of a file:

```
head -c -size_in_bytes filename
```

hostname

Show or set the system’s host name.

- Show current host name:

```
hostname
```

- Show the network address of the host name:

```
hostname -i
```

- Show all network addresses of the host:

```
hostname -I
```

- Show the FQDN (Fully Qualified Domain Name):

```
hostname --fqdn
```

- Set current host name:

```
hostname new_hostname
```

iostat

Report stats for devices and partitions.

- Display disk statistics with disk IDs in human readable format:

```
iostat -h
```

- Display disk statistics with disk names (including LVM) in human readable format:

```
iostat -Nh
```

- Display CPU statistics:

```
iostat -c
```

- Display extended disk statistics with disk names:

```
iostat -xN
```

ip

Show / manipulate routing, devices, policy routing and tunnels.

- List interfaces with detailed info:

```
ip a
```

- Display the routing table:

```
ip r
```

- Make an interface up/down:

```
ip link set interface up/down
```

- Add/Delete an ip address to an interface:

```
ip addr add/del ip/mask dev interface
```

- Add an default route:

```
ip route add default via ip dev interface
```

iptables

Program that allows to configure tables, chains and rules provided by the Linux kernel firewall.

- See chains and rules for specific table:

```
sudo iptables -t table_name -vnL
```

- Set chain policy rule:

```
sudo iptables -p chain rule
```

- Append rule to chain policy for IP:

```
sudo iptables -A chain -s ip -j rule
```

- Append rule to chain policy for IP considering protocol and port:

```
sudo iptables -A chain -s ip -p protocol --dport port -j rule
```

- Delete chain rule:

```
sudo iptables -D chain rule_line_number
```

- Save iptables configuration:

```
sudo iptables-save > path_to_iptables_file
```

jobs

BASH builtin for viewing information about processes spawned by the current shell.

- View jobs spawned by the current shell:

```
jobs
```

- List jobs and their process ids:

```
jobs -l
```

- Display information about jobs with changed status:

```
jobs -n
```

- Display process id of process group leader:

```
jobs -p
```

- Display running processes:

```
jobs -r
```

- Display stopped processes:

```
jobs -s
```

journalctl

Query the systemd journal.

- Show all messages from this boot:

```
journalctl -b
```

- Show all messages from last boot:

```
journalctl -b -1
```

- Follow new messages (like `tail -f` for traditional syslog):

```
journalctl -f
```

- Show all messages by a specific unit:

```
journalctl -u unit
```

- Show all messages by a specific process:

```
journalctl _PID=pid
```

- Show all messages by a specific executable:

```
journalctl /path/to/executable
```

locate

Find filenames quickly.

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

```
locate pattern
```

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

```
sudo updatedb
```

lsattr

List file attributes on a Linux file system.

- Display the attributes of the files in the current directory:

```
lsattr
```

- List the attributes of files in a particular path:

```
lsattr path
```

- List file attributes recursively in the current and subsequent directories:

```
lsattr -R
```

- Show attributes of all the files in the current directory, including hidden ones:

```
lsattr -a
```

- Display attributes of directories in the current directory:

```
lsattr -d
```

lsb_release

Provides certain LSB (Linux Standard Base) and distribution-specific information.

– Print all available information:

```
lsb_release -a
```

– Print a description (usually the full name) of the operating system:

```
lsb_release -d
```

– Print only the operating system name (ID), suppressing the field name:

```
lsb_release -i -s
```

– Print the release number and codename of the distribution, suppressing the field names:

```
lsb_release -rcs
```

lsblk

Lists information about devices.

– List all storage devices in a tree-like format:

```
lsblk
```

– Also list empty devices:

```
lsblk -a
```

– Print the SIZE column in bytes rather than in a human-readable format:

```
lsblk -b
```

– Output info about filesystems:

```
lsblk -f
```

– Use ASCII characters for tree formatting:

```
lsblk -i
```

– Output info about block-device topology:

```
lsblk -t
```

ltrace

Display dynamic library calls of a process.

- Print (trace) library calls of a program binary:

```
ltrace ./program
```

- Count library calls. Print a handy summary at the bottom:

```
ltrace -c /path/to/program
```

- Trace calls to malloc and free, omit those done by libc:

```
ltrace -e malloc+free-@libc.so* /path/to/program
```

- Write to file instead of terminal:

```
ltrace -o file /path/to/program
```

md5sum

Calculate MD5 cryptographic checksums.

- Calculate the MD5 checksum for a file:

```
md5sum filename1
```

- Calculate MD5 checksums for multiple files:

```
md5sum filename1 filename2
```

- Read a file of MD5SUMs and verify all files have matching checksums:

```
md5sum -c filename.md5
```

mdadm

RAID management utility.

- Create array:

```
mdadm --create /path/to/raid_device_file --level  raid level  --raid-devices
number of disks /path/to/disk_device_file
```

- Stop array:

```
mdadm -S /path/to/raid_device_file
```

- Mark disk as failed:

```
mdadm /path/to/raid_device_file -f /path/to/disk_device_file
```

- Remove disk:

```
mdadm /path/to/raid_device_file -r /path/to/disk_device_file
```

- Add disk to array:

```
mdadm /path/to/raid_device_file -a /path/to/disk_device_file
```

- Show RAID info:

```
mdadm -D /path/to/raid_device_file
```

mke2fs

Creates a Linux filesystem inside a partition.

- Create an ext2 filesystem in partition 1 of device b (sdb1):

```
mkfs.ext2 /dev/sdb1
```

- Create an ext3 filesystem in partition 1 of device b (sdb1):

```
mkfs.ext3 /dev/sdb1
```

- Create an ext3 filesystem in partition 1 of device b (sdb1):

```
mkfs.ext3 /dev/sdb1
```

mkfs.cramfs

Creates a ROM filesystem inside a partition.

- Create a ROM filesystem inside partition 1 on device b (sdb1):

```
mkfs.cramfs /dev/sdb1
```

- Create a ROM filesystem with a volume-name:

```
mkfs.cramfs -n volume-name /dev/sdb1
```

mkfs.exfat

Creates an exfat filesystem inside a partition.

- Create an exfat filesystem inside partition 1 on device b (sdb1):

```
mkfs.exfat /dev/sdb1
```

- Create filesystem with a volume-name:

```
mkfs.exfat -n volume-name /dev/sdb1
```

- Create filesystem with a volume-id:

```
mkfs.exfat -i volume-id /dev/sdb1
```

mkfs.fat

Creates an MS-DOS filesystem inside a partition.

- Create a fat filesystem inside partition 1 on device b (sdb1):

```
mkfs.fat /dev/sdb1
```

- Create filesystem with a volume-name:

```
mkfs.fat -n volume-name /dev/sdb1
```

- Create filesystem with a volume-id:

```
mkfs.fat -i volume-id /dev/sdb1
```

- Use 5 instead of 2 file allocation tables:

```
mkfs.fat -f 5 /dev/sdb1
```

mkfs.minix

Creates a Minix filesystem inside a partition.

- Create a Minix filesystem inside partition 1 on device b (sdb1):

```
mkfs.minix /dev/sdb1
```

mkfs.ntfs

Creates a NTFS filesystem inside a partition.

- Create a NTFS filesystem inside partition 1 on device b (sdb1):

```
mkfs.ntfs /dev/sdb1
```

- Create filesystem with a volume-label:

```
mkfs.ntfs -L volume-label /dev/sdb1
```

- Create filesystem with specific UUID:

```
mkfs.ntfs -U UUID /dev/sdb1
```

mkfs.vfat

Creates an MS-DOS filesystem inside a partition.

- Create a.vfat filesystem inside partition 1 on device b (sdb1):

```
mkfs.vfat /dev/sdb1
```

- Create filesystem with a volume-name:

```
mkfs.vfat -n volume-name /dev/sdb1
```

- Create filesystem with a volume-id:

```
mkfs.vfat -i volume-id /dev/sdb1
```

- Use 5 instead of 2 file allocation tables:

```
mkfs.vfat -f 5 /dev/sdb1
```

nethogs

Monitor bandwidth usage per process.

- Start nethogs as root (default device is eth0):

```
sudo nethogs
```

- Monitor bandwidth on specific device:

```
sudo nethogs device
```

- Monitor bandwidth on multiple devices:

```
sudo nethogs device1 device2
```

- Specify refresh rate:

```
sudo nethogs -t seconds
```

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:

```
netstat -p
```

- List information continuously:

```
netstat -c
```

- List routes and do not resolve IP to hostname:

```
netstat -rn
```

- List listening TCP and UDP ports (+ user and process if you're root):

```
netstat -lepunt
```

nm

List symbol names in object files.

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

```
nm -g file.o
```

- Demangle C++ symbols (make them readable):

```
nm --demangle file.o
```

- List only undefined symbols in a file:

```
nm -u file.o
```

- List all symbols, even debugging symbols:

```
nm -a file.o
```

notify-send

Uses the current desktop environment’s notification system to create a notification.

- Show a notification with the title “Test” and the content “This is a test”:

```
notify-send "Test" "This is a test"
```

- Show a notification with a custom icon:

```
notify-send -i icon.png "Test" "This is a test"
```

- Show a notification for 5 seconds:

```
notify-send -t 5000 "Test" "This is a test"
```

pacman

Arch Linux package manager utility.

- Synchronize and update all packages:

```
pacman -Syu
```

- Install a new package:

```
pacman -S package-name
```

- Remove a package and its dependencies:

```
pacman -Rs package-name
```

- Search the package database for a keyword:

```
pacman -Ss icon theme
```

- List installed packages and versions:

```
pacman -Q
```

- List only the explicitly installed packages and versions:

```
pacman -Qe
```

- Find which package owns a certain file:

```
pacman -Qo filename
```

- Empty package cache to free up space:

```
pacman -Scc
```

pkgadd

Add a package to a CRUX system.

- Install a local software package:

```
pkgadd package-name
```

- Update an already installed package from a local package:

```
pkgadd -u package-name
```

pkginfo

Query the package database on a CRUX system.

- List installed packages and their versions:

```
pkginfo -i
```

- List files owned by a package:

```
pkginfo -l package-name
```

- List the owner(s) of files matching a pattern:

```
pkginfo -o pattern
```

- Print the footprint of a file:

```
pkginfo -f file
```

pkgmk

Make a binary package for use with pkgadd on CRUX.

- Make and download a package:

```
pkgmk -d
```

- Install the package after making it:

```
pkgmk -d -i
```

- Upgrade the package after making it:

```
pkgmk -d -u
```

- Ignore the footprint when making a package:

```
pkgmk -d -if
```

- Ignore the MD5 sum when making a package:

```
pkgmk -d -im
```

- Update the package’s footprint:

```
pkgmk -uf
```

pkgrm

Remove a package from a CRUX system.

- Remove an installed package:

```
pkgrm package-name
```

ports

Update/list the ports tree on a CRUX system.

- Update the ports tree:

```
ports -u
```

- List the ports in the current tree:

```
ports -l
```

- Check the differences between installed packages and the ports tree:

```
ports -d
```

print

An alias to a `run-mailcap`’s action print. Originally `run-mailcap` is used to process mime-type/file.

- Print action can be used to print any file on default run-mailcap tool:

```
print filename
```

- With `run-mailcap`:

```
run-mailcap --action=print filename
```

prt-get

The advanced CRUX package manager.

- Install a package:

```
prt-get install package-name
```

- Install a package with dependency handling:

```
prt-get depinst package-name
```

- Update a package manually:

```
prt-get upgrade package-name
```

- Remove a package:

```
prt-get remove package-name
```

- Upgrade the system from the local ports tree:

```
prt-get sysup
```

- Search the ports tree:

```
prt-get search package-name
```

- Search for a file in a package:

```
prt-get fsearch file
```

pwgen

Generate pronounceable passwords.

- Generate random password with symbols:

```
pwgen -y length
```

- Generate hard-to-memorize passwords:

```
pwgen -s length
```

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

```
pwgen -c length
```

reboot

Reboot the system.

- Reboot immediately:

```
reboot
```

- Reboot immediately without gracefully shutdown:

```
reboot -f
```

rpm

RPM Package Manager.

- Show version of httpd package:

```
rpm -q httpd
```

- List versions of all matching packages:

```
rpm -qa 'mariadb*'
```

- Identify owner of a file and show version of the package:

```
rpm -qf /etc/postfix/main.cf
```

- List package-owned files:

```
rpm -ql kernel
```

- Show scriptlets from an RPM file:

```
rpm -qp --scripts some.rpm
```

- Show changed, missing and/or incorrectly installed files of matching packages:

```
rpm -Va 'php-*'
```

run-mailcap

Run MailCap Programs. Run mailcap view, see, edit, compose, print - execute programs via entries in the mailcap file (or any of its aliases) will use the given action to process each mime-type/file.

- Individual actions/programs on run-mailcap can be invoked with action flag:

```
run-mailcap --action=ACTION [--option[=value]]
```

- In simple language:

```
run-mailcap --action=ACTION filename
```

- Turn on extra information:

```
run-mailcap  --action=ACTION --debug filename
```

- Ignore any "copiousoutput" directive and forward output to STD☐OUT:

```
run-mailcap --action=ACTION --nopager filename
```

- Display the found command without actually executing it:

```
run-mailcap --action=ACTION --norun filename
```

screenfetch

Display system information.

- Start screenfetch:

```
screenfetch
```

- Take a screenshot (requires 'scrot'):

```
screenfetch -s
```

- Specify distribution logo:

```
screenfetch -A 'distribution_name'
```

- Specify distribution logo and text:

```
screenfetch -D 'distribution_name'
```

- Strip all color:

```
screenfetch -N
```

see

Alias to `run-mailcap`'s view. An alias to a `run-mailcap`'s action print.

- See action can be used to view any file (usually image) on default mailcap explorer:

```
see filename
```

- Using with `run-mailcap`:

```
run-mailcap --action=view filename
```

sha1sum

Calculate SHA1 cryptographic checksums.

- Calculate the SHA1 checksum for a file:

```
sha1sum filename1
```

- Calculate SHA1 checksums for multiple files:

```
sha1sum filename1 filename2
```

- Read a file of SHA1 sums and verify all files have matching checksums:

```
sha1sum -c filename.sha1
```

sha224sum

Calculate SHA224 cryptographic checksums.

- Calculate the SHA224 checksum for a file:

```
sha224sum filename1
```

- Calculate SHA224 checksums for multiple files:

```
sha224sum filename1 filename2
```

- Read a file of SHA224 sums and verify all files have matching checksums:

```
sha224sum -c filename.sha224
```

sha256sum

Calculate SHA256 cryptographic checksums.

– Calculate the SHA256 checksum for a file:

```
sha256sum filename1
```

– Calculate SHA224 checksums for multiple files:

```
sha256sum filename1 filename2
```

– Read a file of SHA256 sums and verify all files have matching checksums:

```
sha256sum -c filename.sha256
```

sha384sum

Calculate SHA384 cryptographic checksums.

– Calculate the SHA384 checksum for a file:

```
sha384sum filename1
```

– Calculate SHA384 checksums for multiple files:

```
sha384sum filename1 filename2
```

– Read a file of SHA384 sums and verify all files have matching checksums:

```
sha384sum -c filename.sha384
```

sha512sum

Calculate SHA512 cryptographic checksums.

– Calculate the SHA384 checksum for a file:

```
sha512sum filename1
```

– Calculate SHA384 checksums for multiple files:

```
sha512sum filename1 filename2
```

– Read a file of SHA512 sums and verify all files have matching checksums:

```
sha512sum -c filename.sha512
```

shuf

Generate random permutations.

- Randomize the order of lines in a file and output the result:

```
shuf filename
```

- Only output the first n entries of the result:

```
shuf -n n filename
```

- Write output to another file:

```
shuf -o another_filename filename
```

- Generate random numbers in range:

```
shuf -i low-high
```

shutdown

Shutdown and reboot the system.

- Power off (halt) immediately:

```
shutdown -h now
```

- Reboot immediately:

```
shutdown -r now
```

- Reboot in 5 minutes:

```
shutdown -r +5 &
```

- Shutdown at 1:00 pm (Uses 24h clock):

```
shutdown -h 13:00
```

- Cancel a pending shutdown/reboot operation:

```
shutdown -c
```

ss

Utility to investigate sockets.

- Show all TCP/UDP/RAW/UNIX sockets:

```
ss -a -t|-u|-w|-x
```

- Filter TCP sockets by states, only/exclude:

```
ss state/exclude bucket/big/connected/synchronized/...
```

- Show all TCP sockets connected to the local HTTPS port (443):

```
ss -t src :443
```

- Show all TCP sockets along with processes connected to a remote ssh port:

```
ss -pt dst :ssh
```

- Show all UDP sockets connected on specific source and destination ports:

```
ss -u 'sport == :source_port and dport == :destination_port'
```

- Show all TCP IPv4 sockets locally connected on the subnet 192.168.0.0/16:

```
ss -4t src 192.168/16
```

ssh-copy-id

Install your public key in a remote machine's authorized_keys.

- Copy the given public key to the remote:

```
ssh-copy-id -i path/to/certificate username@remote_host
```

- Copy the given public key to the remote with specific port:

```
ssh-copy-id -i path/to/certificate -p port username@remote_host
```

strace

Troubleshooting tool for tracing system calls.

- Start tracing a specific process by its PID:

```
strace -p pid
```

- Trace a process and filter output by system call:

```
strace -p pid -e system call name
```

- Count time, calls, and errors for each system call and report a summary on program exit:

```
strace -p pid -c
```

- Show the time spent in every system call:

```
strace -p pid -T
```

sysctl

List and change kernel runtime variables.

- Show all available variables and their values:

```
sysctl -a
```

- Set a changeable kernel state variable:

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

- Get currently open file handlers:

```
sysctl fs.file-nr
```

- Get limit for simultaneous open files:

```
sysctl fs.file-max
```

- Apply changes from /etc/sysctl.conf:

```
sysctl -p
```

systemctl

Control the systemd system and service manager.

- List failed units:

```
systemctl --failed
```

- Start/Stop/Restart/Reload a service:

```
systemctl start/stop/restart/reload unit
```

- Show the status of a unit:

```
systemctl status unit
```

- Enable/Disable a unit to be started on bootup:

```
systemctl enable/disable unit
```

- Mask/Unmask a unit, prevent it to be started on bootup:

```
systemctl mask/unmask unit
```

- Reload systemd, scanning for new or changed units:

```
systemctl daemon-reload
```

systemd-analyze

Show timing details about the boot process of units (services, mount points, devices, sockets).

- List time of each unit to start up:

```
systemd-analyze blame
```

- Print a tree of the time critical chain of units:

```
systemd-analyze critical-chain
```

tcpflow

Capture TCP traffic for debugging and analysis.

- Show all data on the given interface and port:

```
tcpflow -c -i eth0 port 80
```

timedatectl

Control the system time and date.

- To check the current system clock time:

```
timedatectl
```

- To set the local time of the system clock directly:

```
timedatectl set-time "yyyy-MM-dd hh:mm:ss"
```

- To list available timezones:

```
timedatectl list-timezones
```

- To change timezones:

```
timedatectl set-timezone timezone
```

- To enable Network Time Protocol (NTP) syncing:

```
timedatectl set-ntp on
```

top

Display dynamic real-time information about running processes.

- Start top:

```
top
```

- Start top ignoring any idle or zombie processes:

```
top -i
```

- Start top displaying only processes owned by given user:

```
top -u user-name
```

- Get help about interactive commands:

```
?
```

ufw

Uncomplicated Firewall. Frontend for iptables aiming to make configuration of a firewall easier.

- Enable ufw:

```
ufw enable
```

- Disable ufw:

```
ufw disable
```

- Add ufw allow rule:

```
ufw allow port service_name
```

- Add ufw deny rule:

```
ufw deny port service_name
```

- Show ufw rules:

```
ufw status
```

ulimit

Get and set user limits.

- Get the properties of all the user limits:

```
ulimit -a
```

- Get hard limit for the number of simultaneously opened files:

```
ulimit -H -n
```

- Get soft limit for the number of simultaneously opened files:

```
ulimit -S -n
```

- Set max per-user process limit:

```
ulimit -u 30
```

umask

Manage the read/write/execute permissions that are masked out (i.e. restricted) for newly created files by the user.

- Display the current mask in octal notation:

```
umask
```

- Display the current mask in symbolic (human-readable) mode:

```
umask -S
```

- Change the mask symbolically to allow read permission for all users (the rest of the mask bits are unchanged):

```
umask a+r
```

- Set the mask (using octal) to restrict no permissions for the file's owner, and restrict all permissions for everyone else:

```
umask 077
```

useradd

Create a new user.

- Create new user:

```
useradd name
```

- Create new user with a default home directory:

```
useradd -m name
```

- Create new user with specified shell:

```
useradd -s /path/to/shell name
```

- Create new user with supplementary groups (mind the lack of whitespace):

```
useradd -G group1,group2 name
```

userdel

Remove a user.

- Remove a user and their home directory:

  ```
  userdel -r name
  ```

usermod

Modifies a user account.

- Change a user’s name:

  ```
  usermod -l newname user
  ```

- Add user to supplementary groups (mind the whitespace):

  ```
  usermod -a -G group1,group2 user
  ```

- Create a new home directory for a user and move their files to it:

  ```
  usermod -m -d /path/to/home user
  ```

wall

Write a message on the terminals of users currently logged in. Only available to super-user.

- Send a message:

  ```
  echo "message" | wall
  ```

- Send a message from a file:

  ```
  wall file
  ```

- Send a message with timeout (default 300):

  ```
  wall -t seconds file
  ```

watch

Execute a command repeatedly, and monitor the output in full-screen mode.

- Monitor files in the current folder:

```
watch ls
```

- Monitor disk space and highlight the changes:

```
watch -d df
```

- Monitor “node” processes, refreshing every 3 seconds:

```
watch -n 3 "ps aux | grep node"
```

wpa_cli

Add and configure wlan interfaces.

- Scan for available networks:

```
wpa_cli scan
```

- Show scan results:

```
wpa_cli scan_results
```

- Add a network:

```
wpa_cli add_network number
```

- Set a network’s SSID:

```
wpa_cli set_network number ssid "SSID"
```

- Enable network:

```
wpa_cli enable_network number
```

- Save config:

```
wpa_cli save_config
```

xclip

Copy STDIN to clipboard or print clipboard to STDOUT.

- Copy output to clipboard:

```
echo 123 | xclip -i
```

- Paste clipboard:

```
xclip -o > file.txt
```

xinput

List available input devices, query information about a device and change input device settings.

- List all input devices:

```
xinput list
```

- Disconnect an input from its master:

```
xinput float id
```

- Reattach an input as slave to a master:

```
xinput reattach id master_id
```

xrandr

Set the size, orientation and/or reflection of the outputs for a screen.

- Display the current state of the system (known screens, resolutions, …):

```
xrandr --query
```

- Disable disconnected outputs and enable connected ones with default settings:

```
xrandr --auto
```

- Change the resolution and update frequency of DisplayPort 1 to 1920x1080, 60Hz:

```
xrandr --output DP1 --mode 1920x1080 --rate 60
```

- Set the resolution of HDMI2 to 1280x1024 and put it on the right of DP1:

```
xrandr --output HDMI2 --mode 1280x1024 --right-of DP1
```

- Disable the VGA1 output:

```
xrandr --output VGA1 --off
```

xsetwacom

Command line tool to change settings for Wacom pen tablets at runtime.

- List all the available wacom devices. The device name is in the first column:

```
xsetwacom list
```

- Set Wacom area to specific screen. Get name of the screen with `xrandr`:

```
xsetwacom set "device name" MapToOutput screen
```

- Set mode to relative (like a mouse) or absolute (like a pen) mode:

```
xsetwacom set "device name" Mode "Relative|Absolute"
```

- Rotate the input (useful for tablet-PC when rotating screen) by 0|90|180|270 degrees from "natural" rotation:

```
xsetwacom set "device name" Rotate none|half|cw|ccw
```

- Set button to only work when the tip of the pen is touching the tablet:

```
xsetwacom set "device name" TabletPCButton "on"
```

yaourt

Arch Linux utility for building packages from the Arch User Repository.

- Synchronize and update all packages (including AUR):

```
yaourt -Syua
```

- Install a new package (includes AUR):

```
yaourt -S package-name
```

- Remove a package and its dependencies (includes AUR packages):

```
yaourt -Rs package-name
```

- Search the package database for a keyword (including AUR):

```
yaourt -Ss package-name
```

- List installed packages, versions, and repositories (AUR packages will be listed under the repository name 'local'):

```
yaourt -Q
```

yum

Package management utility for RHEL, Feodra, and CentOS (for older versions).

- Synchronize list of packages and versions available. This should be run first, before running subsequent yum commands:

  ```
  yum update
  ```

- Install a new package:

  ```
  yum install package
  ```

- Install a new package and assume yes to all questions (also works with update, great for automated updates):

  ```
  yum -y install package
  ```

- Remove a package:

  ```
  yum remove package
  ```

- Upgrade installed packages to newest available versions:

  ```
  yum upgrade
  ```

zypper

SUSE & openSUSE package management utility.

- Synchronize list of packages and versions available:

  ```
  zypper refresh
  ```

- Install a new package:

  ```
  zypper install package
  ```

- Remove a package:

  ```
  zypper remove package
  ```

- Upgrade installed packages to newest available versions:

  ```
  zypper update
  ```

- Search package via keyword:

  ```
  zypper search keyword
  ```

3 OSX

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
```

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 plain_text | base64
```

- Decode from stdin:

```
echo base64_text | base64 -D
```

brew

Package manager for OS X.

- Search formula:

```
brew search text
```

- Install formula:

```
brew install formula
```

- List installed formulae [with matching name]:

```
brew list [text]
```

- Get latest version of installed formula (passing no formula updates all installed formulae):

```
brew upgrade [formula]
```

- Update brew:

```
brew update
```

- Switch version of formula:

```
brew switch formula version
```

caffeinate

Prevent a system from sleeping.

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

```
caffeinate -u -t 3600
```

- Prevent mac from sleeping until a command completes:

```
caffeinate -s command
```

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 OS X 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
```

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

Estimate file space usage.

- Get a sum of the total size of a file/folder in human readable units:

```
du -sh file/directory
```

- List file sizes of a directory and any subdirectories in KB:

```
du -k file/directory
```

- Get recursively, individual file/folder sizes in human readable form:

```
du -ah directory
```

- List the KB sizes of directories for N levels below the specified directory:

```
du -k -depth=1 directory
```

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
```

locate

Find filenames quickly.

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

```
locate pattern
```

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

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

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
```

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 (see c++filt).

- 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
```

open

Opens files, directories and applications.

- Open a file with the associated application:

  ```
  open file.ext
  ```

- Run a graphical MacOSX application:

  ```
  open /Applications/Application.app
  ```

- Open the current directory in Finder:

  ```
  open .
  ```

- Reveal a file in finder:

  ```
  open -R /path/dir/file
  ```

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

  ```
  open *.ext
  ```

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
  ```

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 /existing//thumbnail/directory
```

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
```

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."
```

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/'
```

- 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
```

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
```

sw_vers

Print Mac OSX Software versioning information.

- Print OSX Version:

```
sw_vers -productVersion
```

- Print OSX 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 -setusingnetwo
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 -O time
```

- Start top displaying only processes owned by given user:

```
top -user user-name
```

- Get help about interactive commands:

```
?
```

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 Mac OS X 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
```

XCTool

Tool for building Xcode projects.

- Build a single project without any workspace:

```
xctool.sh -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
```

4 SUNOS

devfsadm

Administration command for /dev. Maintains the /dev namespace.

– Scan for new disks:

```
devfsadm -c disk
```

– Cleanup any dangling /dev links and scan for new device:

```
devfsadm -C -v
```

– Dry-run - output what would be changed but make no modifications:

```
devfsadm -C -v -n
```

prctl

Get or set the resource controls of running processes,. Tasks, and projects.

– Examine process limits and permissions:

```
prctl PID
```

– Examine process limits and permissions in machine parseable format:

```
prctl -P PID
```

– Get specific limit for a running process:

```
prctl -n process.max-file-descriptor PID
```

prstat

Report active process statistics.

- Examine all processes and reports statistics sorted by CPU usage:

```
prstat
```

- Examine all processes and reports statistics sorted by memory usage:

```
prstat -s rss
```

- Report total usage summary for each user:

```
prstat -t
```

- Report microstate process accounting information:

```
prstat -m
```

- Print out a list of top 5 cpu using processes every second:

```
prstat -c -n 5 -s cpu 1
```

svcadm

Manipulate service instances.

- Enable a service in the service database:

```
svcadm enable service_name
```

- Disable service:

```
svcadm disable service_name
```

- Restart a running service:

```
svcadm restart service_name
```

- Command service to re-read configuration files:

```
svcadm refresh service_name
```

- Clear a service from maintenance state and command it to start:

```
svcadm clear service_name
```

svccfg

Import, export, and modify service configurations.

- Validate configuration file:

```
svccfg validate smf.xml
```

- Export service configurations to file:

```
svccfg export servicename > smf.xml
```

- Import/update service configurations from file:

```
svccfg import smf.xml
```

svcs

List information about running services.

- List all running services:

```
svcs
```

- List services that are not running:

```
svcs -vx
```

- List information about a service:

```
svcs apache
```

- Show location of service log file:

```
svcs -L apache
```

- Display end of a service log file:

```
tail $(svcs -L apache)
```