Cheat sheet: Tools, commands, shortcuts and hints for Linux

rdaforno, 2020-04

Note: Some of the commands in this document are only available on Ubuntu/Debian based systems.

Basics

Pass / pipe the output of a tool to another (stdout → stdin):	`[tool1] | [tool2]`
Execute two commands on a single line:	`[cmd1]; [cmd2]`
Execute a second command if the first was successful:	`[cmd1] && [cmd2]`
Execute a command in the background:	`[cmd1] &`
Store the output from a command in a file (overwrite):	`[cmd] > [filename]`
Append the output from a command to a file:	`[cmd] >> [filename]`
Feed a file to stdin (standard input) of a tool:	`[cmd] < [filename]`
Ignore output and errors:	`[cmd] > /dev/null 2>&1`
Ignore output and append errors to a log file:	`[cmd] > /dev/null 2>> errors.log`
Open the manual pages for a command:	`man [cmd]`
Run an executable shell script:	`./[scriptname].sh`
Run (interpret) a php script:	`php [filename].php`
Print the system uptime and average load:	`uptime`
Get current date and time:	`date`
Print the current UNIX timestamp:	`date +%s`
Convert a date time string to a UNIX timestamp:	`date "+%s" -d "02/20/2013 08:41:15"`
Show the command history:	`history`
Clear the command history:	`history -c`
Print a string to stdout:	`echo [string]`
Calculate in the terminal:	`echo "(7 + 5) / 2" | bc`
Show list of installed shells:	`cat /etc/shells`
Show current directory (print working directory):	`pwd`

Files and directories

Create a new file:	`touch [filename]`
Copy a file:	`cp [filename1] [dir or filename2]`
Move / rename a file:	`mv [filename1] [dir or filename2]`
Replace a string in all filenames in the current folder:	`rename 's/[searchfor]/[replaceby]/g' *`
Remove a directory (r = recursively, f = force):	`rm -rf [dir]`
Open file explorer from terminal (LinuxMint):	`nemo . &`
Change directory to home:	`cd ~` (note: '~' is optional)
Go back to previous directory:	`cd -`
List directory content (detailed, all files, with full timestamp):	`ls -la --full-time`
... without '.' (current directory) and '..' (parent directory):	`ls -lA`
... sorted by ctime, human readable:	`ls -lth`

... list all files and folders that start with ‘abc’:	`ls -ld abc*`
List folders within the current directory:	`ls -d */`
Create a symbolic link:	`ln -s [targetfile] [linkname]`
Display disk space usage incl. file system type:	`df -h -T`
Show disk usage of current directory:	`du -hsc ./*`
... alternative:	`ncdu`
Find all files ending with "jpg" in current folder and subfolders:	`find -name "*jpg"`
Find all regular files in current folder, ignore subfolders:	`find -maxdepth 1 -type f`
Modify file access rights (o = owner, g = group and e = everyone):	`chmod [oge] [filename]`
... readable and writable (4 + 2) by owner, readable by everyone:	`chmod 644 [filename]`
... full access (rwx = 4 + 2 + 1) for owner, no access for others:	`chmod 700 [filename]`
Make file executable:	`chmod +x`
Add the sticky bit to a folder for the user only:	`chmod u+s`
Recursively change ownership for all files in a folder:	`chmod -R [owner]:[group] [dir]`
Recursively change permissions for all subdirectories:	`find [path] -type d -exec chmod [perm] {} +`
Make all python files in current directory & subfolders executable:	`chmod +x $(find [path] -type f -name *.py)`
Compare files from 2 folders, exclude files starting with ‘.’:	`diff -rq [folder1] [folder2] -x ".*"`
Compare two directories:	`diff -q`
Compress a folder:	`tar -zcvf [outputfile] [dir]`
Extract a compressed file (.tar.gz):	`tar -xvzf [tar filename] -C [outputdir]`
Extract a single file from an archive (.tar.gz):	`tar -xvzf [tar filename] [file in archive]`
Create a 1GB ‘dummy’ file:	`dd if=/dev/zero of=[filename] count=1 bs=1GB`
Generate an md5 checksum for all files in a folder:	`for f in [dir]/*; do md5sum $f; done`
... alternative:	`find [dir] -type f | md5sum`
... alternative:	`md5sum ./*`
Calculate the sha1 checksum of all JPEG files in the a folder:	`find -type f -iname "*.jpg" -exec sha1sum {} \;`
Generate a list of files including the file size:	`find [dir] -printf "%p %s\n"`
... alternative:	`ls -lRA [dir]`
Check the syntax of all php files in a directory:	`find . -iname '*.php' -exec php -l '{}' \;`

Text

Print the contents of a file (write to stdout):	`cat [filename]`
... alternative:	`more [filename]`
Show the first 20 lines of a file:	`head -n 20 [filename]`
Show the last 20 lines of a file:	`tail -n 20 [filename]`
Print only the first 100 characters of each line of a file:	`cat [file] | cut -c1-100`
Read a text file in the terminal:	`less [filename]` (note: use ‘/’ to search)
Edit a text file in the terminal:	`nano [filename]`
Search for a string in a file (basic regex is supported):	`grep [searchfor] [filename]`
Search for a string pattern in a file:	`grep -E [regex] [filename]`
Find all IP addresses in a text file (o = only show matches):	`grep -Eo "([0-9]{1,3}\.){3}[0-9]{1,3}" [filename]`

Filter out lines which contain a certain string (inverted match):
```
grep -v [unwanted_string] [filename]
```

Search in files, recursively (include subfolders, ignore case):
```
grep -i -r [searchfor] [directory]
```

Search a compressed file (zip, gz):
```
zgrep [searchfor] [filename]
```

Replace all occurrences of a string in a text file:
```
sed s/[searchfor]/[replaceby]/g [filename]
```

Replace all occurrences of a string in all c files (in-place):
```
sed -i 's/[searchfor]/[replaceby]/g' *.c
```

Replace all occurrences of a string in all c files, incl. subfolders:
```
find ./ -type f -exec sed -i
 's/[searchfor]/[replaceby]/g' {} \;
```

Replace all tabs with spaces in python files:
```
find ./ -type f -name "*.py" -exec sed -i
 's/\t/    /g' {} \;
```

Delete all lines in a text file that starts with ‘Abc’:
```
sed /^Abc/d [filename]
```

Count the number of lines, words and bytes in a file:
```
wc [filename]
```

GUI tool to compare two text files or two folders:
```
meld [file1 or folder1] [file2 or folder2]
```

Extract the a column from an output (e.g. all PIDs of a user):
```
ps -u [user] | awk '{ printf $1 " " }'
```

Strip non-ASCII characters from a text file:
```
cat [filename] | strings -n 8 > [output]
```

Remove null characters from a text file:
```
cat [filename] | tr -d '\000' > [output_filename]
```

Display whitespace separated data in a tabular way (example):
```
cat /proc/mounts | column -t
```

Count the number of different lines btw two files:
```
diff [file1] [file2] | grep "^>" | wc -l
```

Print the ASCII table:
```
ascii
```

Coding

Display a file in hex format:
```
xxd -s [offset] -l [#bytes] [filename]
```

Display file headers and dump the symbol table of an executable:
```
objdump -f -h -t [exefile]
```

Generate the disassembly from an msp430 executable:
```
msp430-objdump -d [exefile]
```

Compile c code (all warnings enabled):
```
gcc -Wall [files]
```

Version control

SVN Checkout a repository:
```
svn checkout [URL]
```

SVN Update a repository:
```
svn update
```

SVN Submit changes to a repository:
```
svn commit -m [msg]
```

SVN Show a specific revision of a file:
```
svn cat -r [rev.no] [filename]
```

SVN Show the difference between current and an prev. revision:
```
svn diff -r [rev.no] [filename]
```

SVN Show the last changes & rev. no for each line of a file:
```
svn blame [filename]
```

SVN Show changed / unrevised / uncommitted files:
```
svn status
```

SVN Directory listing without checkout:
```
svn ls [url]
```

SVN Move a file (but keep history):
```
svn mv [filename] [newfilename or folder]
```

SVN Display the last 10 log entries (commits):
```
svn log -l 10
```

GIT Clone a repository (download from server):
```
git clone [url]
```

GIT Clone a repository with submodules:
```
git clone [url]; git submodule init; git
 submodule update
```

GIT Add changes to the index (local):
```
git add [filename]
```
(note: use `add -u` to add all)

GIT Rename a file:
```
git mv [filename] [dir]
```

GIT Commit changes to the repository (local):	`git commit -m [title] -m [description]`
GIT Commit changes to the remote location (server):	`git push origin [branch]`
GIT Show last 3 commits:	`git log -n 3`
GIT Compare two commits, ignore whitespaces:	`git diff [sha1 (first 7 chars)] [sha2] -w`
GIT Compare two files from different branches:	`git diff [branch1] [branch2] -- [filename]`
GIT Switch to a different branch:	`git checkout [branch]`
GIT Create a new branch:	`git checkout -b [branch]`
GIT Remove a branch	`git branch -d [branch]`
... and also remove it on the server:	`git push origin --delete [branch]`
GIT Show all branches in the current repository:	`git branch -a`
GIT Show remove URLs:	`git remote -v`
... change the remote URL:	`git remote set-url origin [URL]`
GIT Pull changes from master branch into current branch:	`git pull origin master`
GIT Merge current branch into another:	`git merge [branch]`
GIT Stash local (added, but not yet committed) changes:	`git stash`
... show list of stashes:	`git stash list`
... restore a stash:	`git stash pop`
GIT Discard local changes, restore working tree file:	`git checkout [filename]`
... discard all local changes:	`git checkout .`
GIT Discard / drop local (unpushed) commits:	`git reset --hard @{u}`
GIT Show changes of last commit:	`git show --stat HEAD`
... changed files only:	`git diff HEAD~ --name-only`
GIT Show the last changes for each line of a file:	`git blame [filename]`
GIT Push local tags to the server:	`git push origin --tags`
GIT Add a submodule:	`git submodule add [URL] [directory]`
GIT Change a submodule URL:	`git config submodule.[module_name].url [URL]`
GIT Don’t track file permissions:	`git config core.fileMode false`

Processes, tasks and services

Show most resource intensive processes:	`top` (note: change sort order with < > keys)
List all processes, their parent process IDs and the user:	`ps -ef`
Show the average memory usage of all tasks created by a user:	`ps u -u [user] | awk '{ sum += $4 }; END { printf "%.1f", sum / NR }'`
Print all user processes:	`ps -U [user] -o pid,etime,cmd`
Kill a process:	`kill -9 [process ID]`
Kill all php threads of a user:	`pkill -u [user] php`
Find all PIDs of processes that contain a certain string:	`pgrep -f [searchfor]`
Show the full path of a tool / command:	`which [cmd]`
Locate the binary or source for a command:	`whereis [cmd]`
Restart a service:	`service [service] restart`
... the ‘old’ way:	`/etc/init.d/[service] restart`

... using systemd: `systemctl restart [service]`

Networking

Send 3 pings with packet size 1000 bytes: `ping [hostname or IP] -s 1000 -c 3`

Show open ports (l = listen, t = TCP, u = UDP, 4 = IPv4): `netstat -ltunp4`

Show active TCP connections: `netstat -t`

... alternative: `ss -t`

Perform a TCP SYN port scan: `nmap -sS [host]`

Perform a UDP port scan: `nmap -sU [host]`

Discover hosts in a subnet (ping based): `nmap -sn 192.168.0.0/24`

Resolve domain to IP address and find aliases: `nslookup [domain]`

Resolve an IP address to a hostname (if known): `host [IP]`

Directly ask a DNS server for a specific A record: `dig A @[nameserver] [hostname]`

Save current firewall rules to a file: `iptables-save > [filename]`

Restore stored firewall rules from a file: `iptables-restore < [filename]`

Display all firewall rules: `iptables -S`

List all firewall rules for input chain (verbose): `iptables -vL INPUT`

Clear a chain: `iptables -X [chain]`

Add a new firewall rule (drop incoming TCP packets on port 22):
```
iptables -A INPUT -p tcp --dport 22 -s [IP] \
 -j DROP
```

Allow all connections from localhost (127.0.0.1): `iptables -A INPUT -i lo -j ACCEPT`

Allow incoming packets for already established connections:
```
iptables -A INPUT -m conntrack --cstate \
 RELATED,ESTABLISHED -j ACCEPT
```

Show the network interfaces: `ifconfig`

Bring a network interface down and then up again: `ifdown [interface]; ifup [interface]`

Edit the network config: `nano /etc/network/interfaces`

List local DNS entries: `cat /etc/resolv.conf`

Send an email in the terminal (verbose mode): `echo [body] | mail -v -s [subject] [recipients]`

Run 2 commands on a remote host: `ssh [user]@[host] "[cmd1] && [cmd2]"`

Login to a remote host with X11 window forwarding (GUI): `ssh -X [user]@[host] [cmd]`

Copy files in a folder via ssh to a remote host: `scp [dir]/* [user]@[host]:[path]`

Same as above, but compress files first:
```
tar czf - [dir]/* | ssh [user]@[host]:[path] \
 "tar xvzf -"
```

Record all IP protocol packets going through an interface: `tcpdump -i [interface] -q ip`

Record packets from/to a host from a certain port: `tcpdump -i [interface] ip host [IP] port [port]`

Record packets going to a certain subnet: `tcpdump -i [interface] ip dst net [ip/mask]`

Perform an SNMP walk: `snmpwalk -cpublic -v1 [host]`

Connect to a webserver on port 80: `telnet [server] 80`

Connect to a webserver on port 443: `openssl s_client -connect [server]:443`

UDP dump, capture all UDP packets with a certain destination: `tcpdump -n udp port [port] and dst [dest_ip_addr]`

Send an HTTP POST request with arguments: `curl -s -X POST -d "arg1=val1&arg2=val2" [URL]`

Partitions and devices

List all drive partitions:	`ls /dev/sd*`
Display partitions and their UUID:	`cat /etc/fstab`
Backup/copy a partition (e.g. SD card or USB drive) to a file:	`dd if=/dev/sd[xy] of=[filename]`
Erase/overwrite a partition with zeros:	`dd if=/dev/zero of=/dev/sd[xy]`
Copy a partition to another computer via SSH:	`dd if=/dev/sd[wx] | ssh [user]@[host] "dd of=/dev/sd[yz]"`
Format a partition as EXT3, check for bad blocks in advance:	`mkfs.ext3 -c /dev/sd[xy]`
Commands for assigning a label to a volume / partition:	`e2label`, `mlabel`, `ntfslabel`
Show mounted partitions, formatted as a table:	`mount | column -t`
Mount a partition:	`mount /dev/sd[xy] [mountpoint]`
Show partition UUIDs and types:	`blkid`
List partition tables:	`fdisk -l`
Manipulate a disk partition table:	`fdisk /dev/sd[xy]`
Tools to modify, check and fix partition / partition tables:	`parted`, `gpart`, `gparted` (GUI)
Remount a read-only root file system as read+write:	`mount -o remount,rw /`
Mount a remote partition via SSH:	`sshfs [user]@[host]:/[path] [mountpoint]`
GUI tool to access serial ports:	`gtkterm -p /dev/[device] -s [baudrate]`
Command line tools to access serial ports:	`minicom -D /dev/[device] -b [baudrate]`
... alternative:	`screen /dev/[device] [baudrate]`
List attached USB devices:	`lsusb`
Show I/O stats (disk load):	`iostat`
Monitor disk I/O speed:	`watch -d iostat`
... alternative:	`iotop`
Logical volume manager:	`lvm`
Utility to manage software RAIDs:	`mdadm`
Display infos about physical volumes:	`pvs` `pvdisplay`
Display infos about volume groups:	`vgs` `vgdisplay`
Display infos about logical volumes:	`lvs` `lvdisplay`

Users and groups

Execute a command as another user:	`sudo -u [user] [cmd]`
Switch to another user:	`su [user]`
Print all environment variables:	`printenv`
Show environment variables of another user:	`sudo -Hiu [user] env`
Display username:	`whoami`
Show in which groups a user is:	`groups [user]`
Show logged-in users:	`who`

Change the user login shell:
```
chsh
```

... determine which one is the default shell on a system:
```
ls -l /bin/sh
```

Find out the ID of a user:
```
id [user]
```
(note: add '-u' to get just the UID)

Write a message to a logged-in user on the same machine:
```
write [user] pts/x
```

Change the password for a user:
```
passwd [user]
```

Add a group:
```
addgroup [group]
```

Add a user:
```
adduser [user]
```

Add a user to a group:
```
usermod -a -G [group] [user]
```

... alternative:
```
adduser [user] [group]
```

Formatted list of all users on a system, including their user ID:
```
cat /etc/passwd | \
 awk -F ':' '{print $1 " " $3}' | column -t
```

Packages, scheduling, logs and system info

Install a .deb file:
```
dpkg -i [filename]
```

List installed packages:
```
dpkg -l
```

Purge a package (uninstall and remove all config):
```
apt-get purge [package]
```

Install a new package:
```
apt-get install [package]
```

Simulate package install (shows what would be done):
```
apt-get install --dry-run [package]
```

Search for available packages that contain a keyword:
```
apt-cache search [keyword]
```

Edit crontab for current user:
```
crontab -e
```

Edit crontab of the system:
```
nano /etc/crontab
```

Remove a package from the run level:
```
update-rc.d [package] remove
```

Print the kernel ring buffer (messages):
```
dmesg
```

Follow the content of the system log file:
```
tail -f /var/log/syslog
```

Monitor interrupts with 1s update interval:
```
watch -d -n 1 cat /proc/interrupts
```

Display OS / system info:
```
uname -a
```

... alternative:
```
cat /etc/os-release
```

... alternative:
```
lsb_release -a
```

Print CPU and memory info:
```
lscpu
```

... alternative:
```
cat /proc/meminfo /proc/cpuinfo
```

... delailed hardware infos:
```
dmidecode
```

Store system information as html:
```
sudo lshw -html > sysinfo.html
```

Display sensor readings (temperature, fan speed):
```
sensors
```

Show currently loaded modules:
```
lsmod
```

List open files:
```
lsof
```

Print kernel config:
```
cat /boot/config-$(uname -r)
```

Query the systemd journal:
```
journalctl
```

Insert / remove a kernel module:
```
insmod [module]
rmmod [module]
```

Hints

- All system config files are stored in `/etc`, log files are in `/var/log`. The kernel is kept in `/boot`.
- When moving a root partition to a new harddisk, the file `/etc/fstab` needs to be adjusted. Boot from a live CD and use the bootloader repair tool to reinstall the bootloader or manually reinstall it by executing `grub-install sd[x]`. If your partition is encrypted, you may also need to adjust the `/etc/crypttab`.
- To set the default boot order and timeout, edit the file `/etc/default/grub` and execute `update-grub`.
- There are 6 run levels, the default for a server is 5 (`/etc/rc5.d`). The script `/etc/rc.local` as well as all scripts in `/etc/rcS.d` run at startup, regardless of the run level.
- A '`d`' at the end of a process or binary file often stands for 'daemon' (runs in the background).
- Available sources for `apt` package installer are specified in `/etc/apt/sources.list` and `sources.list.d`.
- One can only switch to a user if the password is set (e.g. not possible for the user `www-data`), but root can always execute a command as that user.
- To permanently add a directory to the `PATH` environment variable, export it in the `~/.bashrc` or `~/.profile` file:
  `export PATH="$PATH:[new_dir]"`
- If a user can't access a device, make sure the user is in the same group (e.g. '`dialout`' or '`tty`').
- To set the language for a user e.g. to English, append the following line to the file `~/.profile`
  `export LANGUAGE="en_US.utf8"`
  To change the default language on a system or generate locales:
  `dpkg-reconfigure locales`
- The home folder for the user `www-data` is typically `/var/www`. For `root` it is `/root`.
- Change root from live Linux into another directory: Mount the partition, bind `dev`, `sys` and `proc` (`mount --bind /dev`) and use `chroot`.
- It is possible to mount a partition as read-only and still overwrite or even format the underlying physical partition (useful to avoid write backs to the disk).
- To make `iptables` changes persistent, simply install the package `iptables-persistent`. Alternatively, one can load the rules when the network interface goes up (place a script with `iptables-restore < [rules_file]` into `/etc/network/if-up.d`).
- To access a remote host with a key rather than a password, insert your public key into the `/home/[user]/.ssh/authorized_keys` file on the remote host (e.g. by using the command `ssh-copy-id`).
- To change the hostname, adjust the files `/etc/hostname` and `/etc/hosts`.
- Install a virtual python environment (requires package `virtualenvwrapper`):
  `mkvirtualenv [my_virtual_environment] --python=`which python3`
  `workon [my_virtual_environment]`
- Install a python module from a local directory / repository:
  `python -m pip install -e .` (or: `pip install -e .`)
- You can remove old kernel versions with `apt-get --purge remove linux-image-[version]; update-grub2`
- To create your own `systemd` service that automatically starts when the system boots, create a service description `/etc/systemd/system/[my-service].service` (permissions 664) with the following content:

```
[Unit]
After=dependency.service
Description=Runs my service
[Service]
Type=idle
ExecStart=/path/to/script.sh
```

```
[Install]
WantedBy=default.target
```

To enable the service:

```
systemctl daemon-reload; systemctl enable [my-service].service
```

Basic shortcuts

- Abort / cancel: `ctrl + c`
- Open new terminal: `ctrl + alt + t`
- Disconnect / close an open connection (e.g. SSH session or terminal): `ctrl + d`
- Minimize all opened windows: `win + d`
- Copy & paste in the terminal: `ctrl + shift + c` and `ctrl + shift + v`
- `Ctrl + z` puts a task into the background. `jobs` prints a list of all user tasks currently in the background and `fg %n` brings jobs #n back to the foreground. `kill %1` will terminate job #1.
- Open a terminal on the login screen: `alt + ctrl + f1`. To switch back to the login screen: `alt + ctrl + f7`.

Regex help

`\`	Escape a special character, required for: `\ ^ . $ | ( ) [ ] * + ? { } ,`
`^`	Match the beginning of the line
`.`	Match any character (except newline)
`$`	Match the end of the line
`|`	OR (alternation)
`()`	Grouping
`[]`	Character class
`*`	Match 0 or more times
`+`	Match 1 or more times
`?`	Match 1 or 0 times
`~`	NOT (negate)
`{n}`	Match exactly n times
`{n,}`	Match at least n times
`{n,m}`	Match between n and m times
`\t`	tab (HT, TAB)
`\n`	newline (LF, NL)
`\r`	`carriage` return (CR)
`\x1B`	hex character
`\w`	Match a word (alphanumeric plus "_", separated by whitespaces)
`\W`	Match a non-word
`\s`	Match any whitespace character
`\S`	Match any non-whitespace character
`\d`	Match a digit character
`\D`	Match a non-digit character