LINUX CHEAT SHEET

by Craciun Dan | v0.4.1 r9 (Sep 11, 2015) | Latest Version

General Purpose Commands

whereis - locate the binary, source and manual page files for a command

whereis searches for binary, source and man pages in standard Linux places. *(Location: /usr/bin/whereis /usr/bin/X11/whereis)*

which - locate a command

which searches for a command in all directories included in PATH. *(Location: /bin/which /usr/bin/which /usr/bin/X11/which)*

apropos - search the manual page names and descriptions

apropos searches for a given pattern in manual page names and descriptions, returning a list of matches. *(Location: /usr/bin/apropos /usr/bin/X11/apropos)*

whatis - display manual page descriptions

whatis displays the name and short description (located in the NAME section of the respective manual page) about a given command. *(Location: /usr/bin/whatis /usr/bin/X11/whatis)*

whoami - print effective userid

whoami displays the username of the currently logged in

file - determine file type

file performs filesystem tests, magic tests and language tests and returns the first match it finds. *(Location: /usr/bin/file /usr/bin/X11/file)*

df - report file system disk space usage

df displays disk usage for all mounted filesystems on the system, showing total size, used size and free space. To see the sizes in human readable form, use **df -h** and to see information for a specific file system only, specify it as an argument to df (e.g. **df -h /dev/sda1**). *(Location: /bin/df)*

du - estimate file space usage

du shows the total size of all the directories, sub-directories and files in the current location. Use the **-h** switch to show human readable sizes. *(Location: /usr/bin/du /usr/bin/X11/du)*

bzip2 - a block-sorting file compressor

bzip2 compresses files offering very good compression sizes. *(Location: /bin/bzip2)*

chgrp - change group ownership

user *(Location: /usr/bin/whoami /usr/bin/X11/whoami)*

id - print real and effective user and group IDs

id displays the current username and the groups it belongs to. *(Location: /usr/bin/id /usr/bin/X11/id)*

cp - copy files and directories

cp copies files and directories. To copy a directory, use **cp -r**. *(Location: /bin/cp)*

dd - convert and copy a file

dd copies a file converting it according to the operands. *(Location: /bin/dd)*

grep - print lines matching a pattern

grep shows all the lines that match a specific given pattern in its input. *(Location: /bin/grep)*

gzip - compress or expand files

gzip compresses files given as arguments reducing size drastically. *(Location: /bin/gzip)*

kill - send a signal to a process

kill sends various signals to processes. The default signal is TERM. *(Location: /bin/kill)*

less - file perusal filter for crt viewing

less is a pager that displays text in a file. It is similar with **more**, but it is more powerful. *(Location: /bin/less)*

ln - make links between files

chgrp changes the group of each given file. *(Location: /bin/chgrp)*

chmod - change file mode bits

chmod changes file and directory permissions, as well as setting file mode bits like the sticky bit. *(Location: /bin/chmod)*

chown - change file owner and group

chown changes the user and/or group ownership of each given file. *(Location: /bin/chown)*

ls - list directory contents

ls lists files and directories as well as information about them. *(Location: /bin/ls)*

mkdir - make directories

mkdir creates directories if they don't already exist. Use **mkdir -p** to create directories recursively (e.g. **mkdir -p $HOME/mydir/mysubdir**). *(Location: /bin/mkdir)*

mv - move (rename) files

mv moves or renames files. *(Location: /bin/mv)*

ps - report a snapshot of the current processes

ps shows the running processes in the current shell. *(Location: /bin/ps)*

pwd - print name of current/working directory

pwd shows the current working directory. *(Location: /bin/pwd)*

rm - remove files or directories

ln creates hard links and symbolic links between files. *(Location: /bin/ln)*

tar - the GNU version of the tar archiving utility

tar stores and extracts files from an archive. *(Location: /bin/tar)*

touch - change file timestamps

touch creates an empty file if it doesn't exist or updates the access and modification time of a file if if already exists. *(Location: /bin/touch)*

uname - print system information

uname shows information about the system, like the kernel version, current date and time, CPU architecture. *(Location: /bin/uname)*

rm removes files or directories. To remove a directory, use **rm -r**. *(Location: /bin/rm)*

mv - move (rename) files

mv moves or renames files. *(Location: /bin/mv)*

sed - stream editor for filtering and transforming text

sed is a powerful utility for manipulating text. *(Location: /bin/sed)*

sort - sort lines of text files

sort is a tool which allows to sort text. *(Location: /usr/bin/sort)*

uniq - report or omit repeated lines

uniq is a tool which will show repeated lines or discard them. *(Location: /usr/bin/uniq)*

Useful One-Liners

Show the Default Shell of the Current User

```
grep $USER /etc/passwd | cut -d ":" -f 7
```

See the Most Used Commands in Bash History

```
history | awk '{print $2}' | awk 'BEGIN {FS="|"}{print $1}
```

Quickly Write a Text File (without an Editor)

```
cat > filename.txt
```

Then type in whatever you like. Press Ctrl+D when you're done. The file **filename.txt** will be overwritten if it exists. Another way to do it:

Show the Currently Running Shell

```
echo $0
```

Or:

```
ps -p $$
```

List All Users Recognized by the System

```
cat /etc/passwd | cut -d ":" -f 1
```

Show the Most Used 20 Commands

```
history | awk '{ print $2 }' | sort | uniq -c | sort -nr |
```

```
cat > filename.txt <<EOF
> input text
> goes here
> EOF
```

Searching for Files

Search Files for a Specific Text

```
find . -iname "*.txt" -exec grep -l "hello" {} +
```

This will search and display all the files ending in **.txt** in the current directory for the text **hello**.

Find Files Modified in the Last N Days

```
find . -iname "*" -mtime -2
```

This will find and display all the files which were modified in the last two days.

Find All Empty Files and Folders

```
find . -iname "*" -empty
```

System Administration

Mount an ISO image

```
sudo mount -o loop /path/to/file.iso /mount/point
```

Mounts file.iso at /mount/point. The mount directory /mount/point should be empty, otherwise the files that it contains will be hidden while the image is mounted (but not lost, they will reappear as soon as the image is unmounted).

Make a Bootable USB Flash Drive from an ISO Image

```
sudo dd bs=4096k if=/path/to/image.iso of=/dev/sdc
```

if stands for input file (the ISO image in this case), while **of** is the USB device, which in this case is */dev/sdc*.

System Configuration

Keyboard Mapping with xev

xev is a small utility which prints contents of X events, so you can assign new key functions to the keyboard using **xmodmap**. Type **xev** to see key events and keycodes. Close the X window to close xev when you're done.

Assign New Keyboard Keys

```
xmodmap -e "keycode 94 = backslash bar"
```

xmodmap can be used to assign to values to keys, so for example pressing the \ key on UK keyboards will have another effect. The above example will make the key to the right of LShift to be **\|** on a UK keyboard.

Tools

Encode FLAC/WAV to Ogg

```
oggenc -b 192 filename.flac
```

You will need to install the **vorbis-tools** package first.

Encode WAV to MP3

```
lame -b 192 filename.wav
```

You will need to install the **lame** package first.

Split FLAC/WAV with CUE

```
cuebreakpoints cue_file.cue | shnsplit audio_file.flac
```

You will need to install the **cuetools** and **shntool** packages first.

Play Movies in Terminal (aaxine)

```
aaxine movie_file.avi
```

You will need to install the **xine-console** package first.

Get HDD Info

```
sudo smartctl -a /dev/sda
```

You will need to install the **smartmontools** package first.

Check HDD Health

Batch Resize JPG Files

```
for i in *.jpg; do convert $i -resize 528x "${i//./_resize
```

Replace **528x** with the desired size in pixels. This specifies the new width, aspect ratio of the original image will be preserved, the original images will be kept and the resized ones will be renamed as **origname_resized.jpg**. You will need to install the **imagemagick** package first.

Create ISO Images from Files/Folders

```
genisoimage -o ouput_file.iso input_directory
```

You will need to install the **genisoimage** package first.

Create ISO Images from CDs/DVDs

```
dd if=/dev/cdrw of=$HOME/output_file.iso
```

Replace **/dev/cdrw** with your device file.

Create ISO Images from Audio CDs

```
cat /dev/cdrw > $HOME/audio_file.iso
```

Replace **/dev/cdrw** with your device file.

Check Filesystem Type (ext3, ext4, etc)

```
df -T
```

```
sudo smartctl -t short /dev/sda
```

After waiting the amount of time specified by the output, use:

```
sudo smartctl -l selftest /dev/sda
```

You will need to install the **smartmontools** package first.

The output will be similar to the output of **df** ran without arguments, but will include an additional column specifying the filesystem type. You can group arguments:

```
df -hT
```

Basic Notions

The Shell

A shell is a command interpreter that can accept commands from the stdin like the keyboard or from a file, called a script. A shell reads command lines, one by one, performs the necessary substitutions, execute the commands and returns the result to the user.

command [option] [argument]

This is the general form of a command, where:

- **command** is the command to execute, usually a program, script or alias located inside directories such as **/bin** and /usr/bin
- **[option]** is an option or group of options to pass to the program; options tell the program how to output or interpret various information (e.g. show or don't show hidden files); options may have a short form (e.g. **-h**) or a long form (e.g. **--human-readable**) and may be grouped together (e.g. instead of **-a -h** you may use **-ah**)
- **[argument]** is the argument given to the program, for example in **ls -l /etc**, -l is an option and /etc is an argument, telling the ls command to list files inside the /etc directory

Most commands (but not all) may be issued without any options or arguments, in which case the program will use its default behavior. For example **ls** issued by itself without any parameters will list the file names in the

Filesystem Hierarchy

The following shows the standard filesystem hierarchy on a Linux system, according to the Filesystem Hierarchy Standard:

- **/bin** - essential user command binaries (e.g. bash, bzip2, cat, chmod, chown, cp, date, df, echo, grep, kill, less, ln, ls, nano, pwd, rm, sed, tar, touch, which, uname)
- **/boot** - static files of the boot loader
- **/dev** - device files
- **/etc** - host-specific system configuration
- **/home** - user home directories (optional)
- **/lib** - essential shared libraries and kernel modules
- **/media** - mount point for removable media
- **/mnt** - mount point for a temporarily mounted filesystem
- **/opt** - add-on application software packages
- **/root** - home directory for the root user (optional)
- **/sbin** - system binaries
- **/srv** - data for services provided by this system
- **/tmp** - temporary files
- **/usr** - user commands, include files, libraries, documentation etc
- **/var** - logs, cache data

In addition to these, most distributions may include the following directories:

- **/proc**
- **/svs**

current working directory, whichever that may be.

Permissions

```
OWNER   GROUP   OTHERS
 rwx     r-x     r-x
 111     101     101
  7       5       5
```

File Types

The first bit in permissions can be:

- **-** for a regular file
- **d** for a directory
- **l** for a symbolic link
- **c** for a special file
- **s** for a socket
- **p** for a named pipe
- **b** for a block device

Basic Commands

ls - list directory contents

This command lists information about files in a directory. It may or may not take options and arguments. For example, **ls** without any arguments will list the file names in the current working directory, while **ls -a /etc** will list all the files inside the /etc directory, including hidden files (preceded by '.') and virtual files (. and ..).

```
$ ls -l /etc
total 1244
drwxr-xr-x  3 root   root      4096 iul 15 12:39 acpi
-rw-r--r--  1 root   root      2981 iul 15 12:30 adduser.c
```

Options like -a or -l can be nested together, like **ls -lh /etc**, which will list the files inside /etc using the long

cd - change the shell working directory

This command changes the current working directory. For example **cd /etc** will change the current working directory to /etc.

```
$ pwd
/home/embryo
$ cd /etc
$ pwd
/etc
$ cd $HOME
$ pwd
/home/embryo
```

In the above example you can see a few examples of using

listing format (-l) and showing human-readable sizes (-h). Several options include:

- **-a, --all** do not ignore entries starting with . (list all files, including the hidden ones)
- **-h, --human-readable** with -l, print sizes in human readable format
- **-X** sort alphabetically by entry extension

(Location: /bin/ls)

cd. After each time the cd command is issued, **pwd** will print the current working directory to reflect the changes. **$HOME** is an environment variable which expands to the home directory of the current user (in this case **/home/embryo**). Without arguments, cd will change the directory to the home directory of the current user. *(Location: **cd** is a Bash builtin)*

Bash Tips

Bash Keyboard Shortcuts

Keyboard shortcuts are very important since they provide fast editing capabilities. They are of really great help when working with the shell. Here is a list of keyboard shortcuts to use in Bash:

- **^F (Ctrl+F)** move cursor one character to the right
- **^B (Ctrl+B)** move cursor one character to the left
- **^A (Ctrl+A)** move cursor to the start of the line
- **^E (Ctrl+E)** move cursor to the end of the line
- **^U (Ctrl+U)** delete all text to the left of the cursor
- **^K (Ctrl+K)** delete all text to the right of the cursor
- **^P (Ctrl+P)** bring up the previous command in history
- **^N (Ctrl+N)** bring up the next command in history
- **^H (Ctrl+H)** delete one character to the left
- **^L (Ctrl+L)** clear the terminal
- **^R (Ctrl+R)** reverse search
- **^C (Ctrl+C)** end a running program
- **^Z (Ctrl+Z)** suspend a running program
- **^D (Ctrl+D)** exit the current shell
- **Alt+F** move cursor one word to the right
- **Alt+B** move cursor one word to the left
- **Tab** command or filename completion

Start Bash in Debug Mode

```
bash -x SCRIPT.sh
```

Print the Remaining Arguments of a Script Starting at a Specified Position

```
echo "${@:N}"
```

Will echo all the remaining arguments passed to a script, starting with Nth argument. Take the following script, called **script.sh**:

```
#!/bin/bash

echo "${@:3}"
```

If ran as **./script.sh ab cd ef gh ij kl**, the output will be:

```
ef gh ij kl
```

Floating-Point Arithmetic Examples

```
echo "5/2" | bc -l
```

```
echo | awk '{ print 5/2 }'
```

```
perl -e 'print 5/2'
```

Bash - Parameter Expansion Tricks

Remove File Extensions

Takes the form **${VARIABLE%PATTERN}** and will remove the first occurence of PATTERN, starting at the end of the string:

```
myfile="abc.txt"
echo ${myfile%.txt}
```

Replace a Substring with Another String

Using **${VARIABLE/PATTERN/STRING}** will replace the first occurence of PATTERN from within the variable with STRING, while **${VARIABLE//PATTERN/STRING}** will replace all occurences:

```
var="apples and oranges"
echo ${var/apples/cherries}
```

Manipulating Paths and Filenames

Print only the filename (without the extension, whichever that may be):

```
var="my_filename.txt"
echo ${var%.*}
```

Print only the filename extension:

```
var="my_filename.txt"
echo ${var#*.}
```

Convert Uppercase to Lowercase

Use **${VARIABLE,,}**:

```
var="ABCDEF"
echo ${var,,}
```

Convert Lowercase to Uppercase

Use **${VARIABLE^^}**:

```
var="abcdef"
echo ${var^^}
```

Remove a Substring from a String

Use **${VARIABLE/PATTERN/}**:

```
var="apples and oranges"
echo ${var/apples/}
```

Print All Arguments Given to a Script Starting at a Specified Position

Use **${@:N}**:

```
./myscript.sh arg1 arg2 arg3 arg4 arg5
echo ${@:3}
```

Output - the arguments will be separated by blanks:

Print only the filename from an absolute path:

```
var="/usr/bin/emacs"
echo ${var##*/}
```

Print only the path, without the filename:

```
var="/usr/bin/emacs"
echo ${var%/*}
```

Bash Builtins

((expression))

Evaluate expression value.

.

Execute commands from a file in the current shell.
Example:

```
. $HOME/.bashrc
```

:

Null command. No effect, the command does nothing.

[

Evaluate conditional expression. This command is the same as the **test** builtin, but the last argument must be a a **]** character to match the opening **[**. Example:

```
if [ -f /bin/bash ]; then
  echo "File /bin/bash exists."
fi
```

```
arg3 arg4 arg5
```

history

Display or manipulate the history list.

if

Execute commands based on conditional. Example:

```
if [ $VAR -gt 10 ]; then
  echo "$VAR is greater than 10."
elif [ $VAR -lt 10 ]; then
  echo "$VAR is less than 10."
else
  echo "$VAR is 10."
fi
```

jobs

Display status of jobs.

kill

Send a signal to a job. The following commands do the same thing, sending the SIGKILL signal to the process with the

```
```

[[

Execute conditional command.

alias

Define or display aliases. Example:

```
alias rmf='rm -f'
```

bg

Move jobs to the background.

bind

Set Readline key bindings and variables.

break

Exit for, while or until loops.

builtin

Execute shell builtins.

caller

Return the context of the current subroutine call.

case

Execute commands based on pattern matching.

PID of 1550:

```
kill -9 1550
kill -SIGKILL 1550
kill -KILL 1550
```

let

Evaluate arithmetic expressions.

Other Commands

qdbus: Show Amarok Metadata Info

```
```

```
qdbus org.kde.amarok /Player GetMetadata
```

qdbus: Change Amarok Volume

```
qdbus org.kde.amarok /Player VolumeSet 40
```

qdbus: Play/Pause Amarok

```
qdbus org.kde.amarok /Player PlayPause
```

Ubuntu/Mint Useful Tips and One-Liners

APT: Upgrade the System

```
sudo apt-get update && sudo apt-get dist-upgrade
```

APT: Add a PPA Repository

```
sudo add-apt-repository ppa:USERNAME/PPA_NAME
```

APT: Show Package Info

```
apt-cache show PACKAGE
```

APT: Clean Up Package Cache

```
sudo apt-get clean
```

APT: Clean Up Packages No Longer Available

```
sudo apt-get autoclean
```

APT: Install a Program's Dependencies

```
sudo apt-get build-dep <PACKAGE>
```

The source repositories (lines starting with deb-src inside your /etc/apt/sources.list file) should be enabled.

APT: Search for Packages by Pattern

DPKG: Install a DEB Package

```
sudo dpkg -i PACKAGE.deb
```

DPKG: Remove a Manually Installed DEB Package

```
sudo dpkg -r PACKAGE.deb
```

DPKG: Forcibly Remove an Installed Package

```
sudo dpkg --purge --force-all PACKAGE
```

DPKG: List All Installed Packages

```
dpkg --get-selections
```

This will list all the packages installed on a system using APT.

DPKG: List Packages That Install a Certain File

```
dpkg -S <FILENAME>
```

This will list all packages that will install **FILENAME**. The pattern **FILENAME** will be matched for files containing it in their name.

DPKG: List Installed Files by a Package

```
apt-cache search <PATTERN>
```

This will search for packages which contain **PATTERN** in their name or description.

APT: List Dependencies of a Package

```
apt-cache depends <PACKAGE>
```

This will list the package names on which **PACKAGE** depends on.

APT: List All Packages that Depend on a Package

```
apt-cache rdepends <PACKAGE>
```

This will list the package names which depend on **PACKAGE**.

APT: Fix Broken Dependencies

```
sudo apt-get -f install
```

```
dpkg -L PACKAGE
```

DPKG: List Contents of a DEB Package

```
dpkg -c <FILE>
```

DPKG: Show the Control File of a DEB Package

```
dpkg -f <FILE>
```

Get a List of Every Installed Package

```
dpkg -l | tr -s ' ' '#' | cut -f2 -d"#"
```

Show Mint Release Info

```
lsb_release -a
```

GSettings: Disable Overlay Scrollbars in Ubuntu

```
gsettings set com.canonical.desktop.interface scrollbar-mo
```

Standard C

malloc()

```
#include <stdlib.h>

void *malloc (size_t size);
```

srand()

```
#include <stdlib.h>

void srand (unsigned int seed);
```

gettimeofday()

fprintf()

```
#include <stdio.h>

int fprintf (FILE *fd, const char *format, ...);
```

fscanf()

```
#include <stdio.h>

int fscanf (FILE *fd, const char *format, ...);
```

sscanf()

```
#include <sys/time.h>

int gettimeofday(struct timeval *tv, struct timezone *tz);
```

fgetc()

```
#include <stdio.h>

int fgetc(FILE *stream);
```

This function reads the next character from **stream** and returns it as an unsigned char cast to an int, or **EOF** on end of file or error.

```
while ((c = fgetc(stream)) != EOF) {
        array[i] = c;
        i++;
}
```

fgets()

```
#include <stdio.h>

char *fgets(char *s, int size, FILE *stream);
```

This function reads at most one less than **size** characters from **stream** and stores them into the buffer pointed to by **s**. Reading stops after an **EOF** or a **newline**. If a newline is read, it is stored into the buffer. A terminating null byte ('**\0**') is stored after the last character in the buffer.

```
#include <stdio.h>

int sscanf (const char *str, const char *format, ...);
```

This function reads the input from the string point to by **str** and formats it according to **format**.

```
char str[5] = "1234";
int n;

if (sscanf(str, "%d", &n) != EOF) {
        fprintf(stdout, "%d", n);
}
```

strcpy()

```
#include <string.h>

char *strcpy(char *dest, const char *src);
```

This function copies the string pointed to by **src**, including the terminating null byte ('**\0**'), to the buffer pointed to by **dest**. The destination string **dest** must be large enough to receive the copy.

strcmp()

```
#include <string.h>

int strcmp(const char *s1, const char *s2);
```

This function compares two strings and returns an integer less than zero if **s1** is found to be less than **s2**, equal to zero if **s1** is found to match **s2**, or greater than zero if **s1** is greater than **s2**.

System Calls

open()

```
#include <fcntl.h>

int open (const char *name, int flags);
int open (const char *name, int flags, mode_t mode);
```

This function is used to open a file. The **flags** argument can be one of **O_RDONLY**, **O_WRONLY** or **O_RDWR**. The **flags** argument can be bitwise-ORed with one or more of several other values, like **O_APPEND**, **O_TRUNC** or **O_CREAT**. If **O_CREAT** is specified, the **mode** argument is also required.

```
fd = open("filename.txt", O_RDONLY | O_CREAT, 0644);
if (fd < 0) {
  perror("ERROR: open()");
}
```

close()

```
#include <unistd.h>

int close (int fd);
```

select()

```
#include <sys/time.h>
#include <sys/types.h>
#include <unistd.h>

int select (int nfds,
            fd_set *read-fds,
            fd_set *write-fds,
            fd_set *except-fds,
            struct timeval *timeout);
```

read()

```
#include <unistd.h>

ssize_t read (int fd, void *buffer, size_t size);
```

write()

```
#include <unistd.h>

ssize_t write (int fd, const void *buffer, size_t size);
```

perror()

```
#include <stdio.h>

void perror (const char *message);
```

```
FD_ZERO (fd_set *set);
FD_SET (int fd, fd_set *set);
FD_CLR (int fd, fd_set *set);
FD_ISSET (int fd, const fd_set *set);
```

This cheatsheet is still **work in progress**. Locations are given for a Linux Mint 17 system, but are mostly the same on all modern distributions.

You can get the latest version of this file from here.

All the feedback is welcome. You can submit suggestions or corrections regarding this document by leaving a comment here or by sending me an email.