Linux Cheat Sheet

Spyros Alertas

Created On: September 12, 2023
Last Edit: October 29, 2023

Contents

1 Introduction

This document is a brief introduction to the basics of the linux universe and some of the most common and useful linux programs.

What is Linux?

- Linux and Unix are Operating Systems.
- Unix is a proprietary OS developed by AT&T Labs.
- Linux is an Open Source OS based on Unix developed by Linus Torvald in 1991.
- Unix OS: MacOS, Solaris, EulerOS, etc.
- Linux Distributions: Ubuntu, Debian, Fedora, OpenSuse, Arch Linux, Gentoo. ElementaryOS.
- The heart of Linux is the Kernel.

What is an Operating System?

- The main purpose of an Operating System is to provide a layer for other Software to communicate with the hardware through API (Application Programming Interface).
- Once loaded into a computer, the Operating System manages all other application programs.

What is the Kernel in Linux?

- The main purpose of an Operating System is to provide a layer for other Software to communicate with the hardware through API (Application Programming Interface).
- Once loaded into a computer, the Operating System manages all other application programs.
- The Kernel is the heart of Linux, it is the interface between the hardware and its processes. It is responsible for managing resources as efficiently as possible.

2 Important Linux Commands Summary - In One Page

Command	Description	Command	Description
pwd	Print working directory	env	Print environment variables
cd	Change directory	export	Create or update environment variable
ll	List directory contents	unset	Delete environment variable
mkdir	Create directory	env | grep var	Search for specific environment variable
rmdir	Delete directory	ssh	Secure Socket Shell - Remote connection
rm	Delete files or directories	scp	Secure Copy Protocol - File transfer
chmod	Change file permissions		
grep	Search for pattern in files		
find	Search for files or directories		
cksum	Calculate the checksum of a file	echo	Print arguments to standard output (stdout)
cat	Display file content	head	Display first N lines only
less	Display file content in terminal	tail	Display last N lines only
sort	Display file content sorted		
uniq	Display uniq or duplicate lines only		

Important linux tools/terminology:

- | : The pipe symbol passes the output of the command on the left as input to the command on the right
- | xargs : The pipe symbol combined with the word xargs will pass the output of the command on the left as argument to the right command
- > and >> : Input/Output redirection. Will change input for the command from a file or the output of the command to a file.
- Absolute Path: Starts from root directory and gives full path.
- Relative Path: Starts from current working directory.
- Wildcard * : Matches with zero or more any characters.
- Wildcard ? : Matches exactly one any character.

3 Basic Linux Commands - Paths, Directories, Files and Permissions

3.1 Relative vs Absolute Paths

- An **Absolute Path** shows the location of a file or directory from the root path.
- A **Relative Path** describes the location of a file or directory from the current location.

3.2 man, arguments (--help, --version), pwd, which, type, ls, ll, cd, clear, whoami

Some basic Linux commands and basic arguments.

man: Displays the manual page of a program.

Example use:

```
spyros@spyros-ubuntu-23:~$ man man
$ example output: manual page of man program

spyros@spyros-ubuntu-23:~$ man pwd
$ example output: manual page of pwd program
```

Arguments: Are options that can be used while running a program to define its behavior.
--help or -h: Displays a help page with info about what the program does and available arguments.
--version or -v: Displays the version of a program.

Example use:

```
spyros@spyros-ubuntu-23:~$ man --help
$ example output: help page of man program

spyros@spyros-ubuntu-23:~$ man --version
$ example output: man 2.1.12 (version of man program)

spyros@spyros-ubuntu-23:~$ man -h
$ example output: help page of man program

spyros@spyros-ubuntu-23:~$ man -v
$ example output: man 2.1.12 (version of man program)
```

pwd: Print Working Directory

Example use:

```
spyros@spyros-ubuntu-23:~$ pwd
$ example output: /home/spyros (current absolute path you are at)
```

which: Locate a command.

Example use:

```
spyros@spyros-ubuntu-23:~$ which ls
$ example output: /usr/bin/ls

spyros@spyros-ubuntu-23:~$ which pwd
$ example output: /usr/bin/pwd
```

type: Tells if a command is a built-in shell command and where it is located or if it is an alias.

Example use:

```
spyros@spyros-ubuntu-23:~$ type mkdir
$ example output: mkdir is /usr/bin/mkdir

spyros@spyros-ubuntu-23:~$ which cd
$ example output: cd is a shell builtin
```

ls and **ll:** List Directory Contents.

Example use:

```
spyros@spyros-ubuntu-23:~$ type ll
$ example output: ll is aliased to `ls -alF'

spyros@spyros-ubuntu-23:~$ ls
$ example output: Displays list of  contents of current directory

spyros@spyros-ubuntu-23:~$ ls -a
$ example output: Include files that start with .
$ (these are hidden files in Linux)

spyros@spyros-ubuntu-23:~$ ls -l
$ example output: Display full details: permissions, group, owner,
$ size and timestamp

spyros@spyros-ubuntu-23:~$ ls -lh
$ example output: Print sizes like 1K, 234M, 2G, etc. Must be
$ combined with -l or -s argument

spyros@spyros-ubuntu-23:~$ ls -t
$ example output: Order by update time

spyros@spyros-ubuntu-23:~$ ls -r
$ example output: Display directory contents in reverse order

spyros@spyros-ubuntu-23:~$ ls -R
```

```
$ example output: Display directory contents and also the content
$ of its subdirectories

spyros@spyros-ubuntu-23:~$ ls -S
$ example output: Display directory contents order by file size -
$ largest number first
```

cd: Change Directory.

Example use:

```
$ Change Directory Using Relative Path
spyros@spyros-ubuntu-23:~$ cd Downloads/mydir
$ example output: spyros@spyros-ubuntu-23:~/Downloads/mydir$

$ Change Directory Using Absolute Path
spyros@spyros-ubuntu-23:~$ cd /home/spyros/Downloads/mydir
$ example output: spyros@spyros-ubuntu-23:~/Downloads/mydir$

$ Notice the difference between relative and absolute paths
$ Absolute Paths start with / while relative cannot

$ Parent Directory
spyros@spyros-ubuntu-23:~$ cd ..
or
spyros@spyros-ubuntu-23:~$ cd ../

$ Two Directories Up
spyros@spyros-ubuntu-23:~$ cd ../../
$ The / can be omitted only from the end, not in between
```

clear: Will clear all output from current terminal window.

Example use:

```
spyros@spyros-ubuntu-23:~$ clear
```

whoami: Will print users effective name.

Example use:

```
spyros@spyros-ubuntu-23:~$ whoami
$ output: spyros
```

3.3 File/Directory Management and Permissions: touch, mkdir, rmdir, rm, chmod

```
$ touch will update access and modification date of file to current time
```

```
$ If file does not exist it will be created
$ -c: don't create file if it doesn't exist
$ -a: change only access time
$ -m: change only modification time
spyros@spyros-ubuntu-23:~$ touch myfile.txt

spyros@spyros-ubuntu-23:~$ mkdir test
$ Will create directory test

spyros@spyros-ubuntu-23:~$ mkdir dir1 dir2
$ Will create two directories with names dir1 and dir2

spyros@spyros-ubuntu-23:~$ rmdir dir1 dir2
$ Will delete directories dir1 and dir2
$ Note: rmdir will only delete empty directories, if there are
$ files it will fail and can delete only directories, not files

$ Command rm can be used to delete both files and directories
$ -d: will delete empty directory (like rmdir)
$ -i: prompt before removal
$ -r or -R: Delete directories and their contents recursively
spyros@spyros-ubuntu-23:~$ rm -r test
$ Will delete test directory even if it has files

spyros@spyros-ubuntu-23:~$ rm myfile.txt
$ Will delete file myfile.txt

$ chmod is used to change permissions to files and directories
spyros@spyros-ubuntu-23:~$ chmod 755 myfile.txt
$ Will change permissions of file to 755
$ Let's understand in next subsection Linux Permissions better
```

3.4 Linux Permissions - chmod xxx

```
spyros@spyros-ubuntu-23:~$ ls -l
drwxrwxr-x       2      spyros  spyros  4096    Sep 15 06:51 album/
-rw-rw-r--       1      spyros  spyros    37    Sep 15 07:23 notes.txt
```

The first field in above output of ls -l is information about file permissions.

- File Type: is the first character

  'd' is for directory and '-' for normal files

- Permissions: rw-rw-r--

  This is actually three sets of permissions

rw- : Owner Permissions - first set

rw- : Group Permissions - second set

r-- : Others Permissions - third set

Every file in linux belongs to an owner and a group

The characters in above strings show which permissions are granted to user/group/others

'r' stands for read, 'w' for write and 'x' for execute

- User Owner: spyros
- Group Owner: spyros

  Every user in linux belongs to a group and many users can belong to one group
- Others: By others we mean users that don't belong in the same group as the owner of a file or directory

With command chmod we set the permissions for each group with octal values (values from 0 to 7).

- 0: no permissions granted
- 1: only execute permission
- 2: only write permission
- 3: execute and write permissions
- 4: only read permission
- 5: read and execute permissions
- 6: read and write permissions
- 7: all (read, write and execute) permissions

```
spyros@spyros-ubuntu-23:~$ chmode 775 test_script.sh
-rwxrwxr-x       1      spyros  spyros    37    Sep 15 07:23 notes.txt

spyros@spyros-ubuntu-23:~$ chmode 664 test_script.sh
-rw-rw-r--       1      spyros  spyros    37    Sep 15 07:23 notes.txt

spyros@spyros-ubuntu-23:~$ chmode 666 test_script.sh
-rw-rw-rw-       1      spyros  spyros    37    Sep 15 07:23 notes.txt
```

4 Basic Linux Commands - Checksum, Search files/directories/file content, Pipe and Redirect

4.1 Checksum - cksum

```
spyros@spyros-ubuntu-23:~$ cksum file.txt
$ output: 1078369332    48      file.txt

$ Computes and verifies file checksums
$ First number is checksum, second is the filesize and
$ third column is the file name

$ File checksum is very usefull when needed to check if
$ a file is tampered or not compared to another version
$ of the same file
```

4.2 Search for files and directories - find

find is a very useful command to search for files and directories.

```
spyros@spyros-ubuntu-23:~$ find dirname -type d
$ output: Find all directories recursively in all
$ subdirectories that match pattern dirname

spyros@spyros-ubuntu-23:~$ find filename -type f
$ output: Find all files recursively in all subdirectories
$ that match pattern filename

spyros@spyros-ubuntu-23:~$ find filename
$ output: Find all files and directories recursively in all
$ subdirectories that match pattern filename
```

4.3 Search file content - grep

grep is a very powerful tool that allows you to search in files for patterns. grep

```
spyros@spyros-ubuntu-23:~$ grep text filename
$ output: Will print the lines that contain pattern text in
$ file with name filename

spyros@spyros-ubuntu-23:~$ grep text *
$ output: Will print the lines and filenames that contain pattern text
$ * means search in all files in current directory

$ Useful options:
$ -i : ignore case (lower or upper case)
```

```
$ -v : invert the sense of matching

$ -o : Print only matching part of the line, not the whole line
$ -n : Print also the number of line that pattern is found at

$ -c : count the number of lines that contain the pattern

$ -l : Print only filenames in which matches are found
$ -L : Print only filenames in which matches are not found

$ -r : Search in all files even in sub-directories, doesn't follow
$ symbolic links unless they are on the command line
$ -R : Same as -r but will follow symbolic links also
```

4.4 Pipe

In Linux we can pipe the output of one command as input stream to another using this symbol |. A pipe passes the standard output from one command as input stream to another command.

```
spyros@spyros-ubuntu-23:~$ commandA | commandB
```

4.5 Pipe with xargs

In Linux we can pipe the output of one command as argument to another using this symbol | xargs. A pipe combined with xargs will pass the standard output from one command as argument to another command.

```
spyros@spyros-ubuntu-23:~$ commandA | xargs commandB
```

4.6 Redirect

In Linux we can redirect the output of one command to a file. We can do it using the symbols > or >>.
> Will create the file if it doesn't exist but will overwrite any content if it's existing file.
>> Will create the file if it doesn't exist but will append at the end of the file the new content, without over writing previous content.
It can also be used to pass a file as input stream to a command.

```
spyros@spyros-ubuntu-23:~$ ls -l > output.txt

spyros@spyros-ubuntu-23:~$ ls -l >> output.txt

spyros@spyros-ubuntu-23:~$ input.txt > command
```

4.7 Pipe vs Pipe with xargs vs Redirect

These three tools may seems similar at first but they are very different. In short:

- Pipe | (commandA | commandB) : output of commandA will be passed as input stream to commandB.
- Pipe with xargs | xargs (commandA | xargs commandB) : output of commandA will be passed as argument to commandB.
- Redirect > or >> (commandA > output-file) or (input-file > commandB) : Output of commandA will be redirected to output-file instead of terminal and input for commandB will be redirected to input-file instead of terminal input.

5 Environment Variables - env, echo, export, unset

```
spyros@spyros-ubuntu-23:~$ env
spyros@spyros-ubuntu-23:~$ printenv
$ These two commands will print all environment variables

spyros@spyros-ubuntu-23:~$ env | grep SESSION
$ This command will print all environment variables that contain SESSION

$ This symbol | is called pipe and it passes the output
$ of the command on the left (in this case env) to the
$ command on the right (here grep SESSION)

spyros@spyros-ubuntu-23:~$ echo $GDMSESSION
$ Echo prints the output of its argument, in this case the value of the
$ environment variable GDMSESSION

spyros@spyros-ubuntu-23:~$ echo Hello Linux World!
$ output: Hello Linux World!

# export name=value
spyros@spyros-ubuntu-23:~$ export myvar="This is my first environment variable"
$ This will create a new or update the value of an existing environment variable
spyros@spyros-ubuntu-23:~$ echo $myvar
# Will print the value of new env variable

spyros@spyros-ubuntu-23:~$ unset myvar
# Will delete environment variable if it exists
spyros@spyros-ubuntu-23:~$ echo $myvar
# Empty line will be printed after unsetting environment variable
```

6 Remote Connect To Linux Machines

6.1 ssh and scp

ssh stands for **Secure Shell** or **Secure Socket Shell** and it's a protocol that allows secure connection to remote Linux machines through text based user interface.
Purpose: Securely connect to remote Linux and run commands. Cannot transfer files from one Linux to another.

scp stands for **Secure Copy Protocol** and it's a file transfer protocol. scp is using ssh for data transfer and security.
Purpose: Securely copy files from one Linux machine to another. Cannot run other Linux commands.

```
spyros@spyros-ubuntu-23:~$ ssh user@host
$ The above will request password if required and after
$ will connect to remote Linux.

spyros@spyros-ubuntu-23:~$ scp source user@host:source/filepath
$ The above will copy source file or directory on location
given after user@host: path to the place where you are running
$ scp command from

$ -p: argument can be passed with port if we don't want
$ default port to be used
```

6.2 Putty and WinSCP

Putty is a terminal emulator that supports ssh protocol.

WinSCP is a Windows GUI that uses **SFTP - Secure File Transfer Protocol** and allows to securely transfer files from remote Linux machines.

- Putty - Official Site
- WinSCP - Official Site

7 More Linux Commands - wc, cat, less, more, sort, uniq, head, tail

```
spyros@spyros-ubuntu-23:~$ ls -l pattern | wc -l
$ output: ls -l will find the files matching pattern
$ then wc -l will count the lines (the number of files)

$ -c : byte counts
$ -m : character counts
```

```
$ -l : line counts
$ -L : length of longest line
$ -w : word counts

spyros@spyros-ubuntu-23:~$ cat filename
$ output: will display the whole file content at once

spyros@spyros-ubuntu-23:~$ less filename
spyros@spyros-ubuntu-23:~$ more filename
$ output: will display file content in pages
$ less is better than more, especially for big files,
$ as it doesn't load the entire file at once

spyros@spyros-ubuntu-23:~$ sort filename
$ output: will display file content sorted in ascending alphabetical order

$ Default sorting is ascending
$ -r : reverse order (descending)
$ -R : random sort
$ -u : unique values only

$ Note: uniq considers repeated lines only if they are consecutive,
$ so it's a good idea to first use command sort
spyros@spyros-ubuntu-23:~$ sort filename | uniq -c
$ will prefix lines by the number of occurences

$ -d : will only display duplicate lines (only once)
$ -D : will display only duplicate lines (as many times as they occur)

$ -u : Only display unique lines

$ -w : compare only N characters, not the whole line

spyros@spyros-ubuntu-23:~$ head -n 2 filename
$ Display only first 2 lines

spyros@spyros-ubuntu-23:~$ tail -n 2 filename
$ Display only last 2 lines
```

8 Advanced Linux Commands - Combination of commands

8.1 Extract content of certain xml tags

In certain cases you may have large xml files but you need all the values of one certain xml tag only. Below command can extract the distinct values of a certain xml tag.

```
spyros@spyros-ubuntu-23:~$ grep -hr "<xmltag" filename |
sed -e 's/<[^>]*>//g' | awk '$1 = $1' | uniq -u |
awk '$0="\047"$0"\047"' | sed -z 's/\n/,/g;s/,$/\n/'
```

Above command explained part by part.

- grep -hr ”<xmltag” filename

  Finds all lines that contain the text <id. Change id as required

- sed -e 's/<[^>]*>//g'

  Removes all xml tags and keeps only its content

- awk '$1 = $1'

  Remove trailing white spaces

- uniq -u (alternatively $\underline{sort - u}$ can achieve the same result but also sort content)

  Removes duplicate lines from output

- awk '$0="\047"$0"\047"'

  Append as prefix and post-fix symbol '.

- sed -z 's/\n/,/g;s/,$/\n/'

  Change lines to a comma separated list

8.2 Find start and end line and copy in between content to new file

This command will search for first and second string, will keep their line numbers and after it will copy all content in between these two lines to a new file.

```
spyros@spyros-ubuntu-23:~$ grep -n "starttext\|endtext" filename |
cut -f1 -d : | head -2 | sed -z 's/\n/,/g;s/,$/\n/' |
xargs -I {} sed -n '{}p' filename > outputfilename
```

Above command explained part by part.

- grep -n "starttext\|endtext" filename

  Finds the lines where starttext or endtext exists.

- cut -f1 -d :

  Deletes everything after the line number

- head -2

  Keeps only the first two results. Just in case they exist more than once

- sed -z 's/\n/,/g;s/,$/\n/’

  Change lines to a comma separated list

- xargs -I sed -n '{}p' filename > outputfilename

  Will copy from file filename to outputfilename all lines that are passed as argument. -I tells xargs to replace with the output of the previous command.

  Note: If you use symbol > it will delete file if it exists and recreate it only with this content. If you change > to >> it will create the file if it does not exist or will append to the end of it, if it already exists.

9 Java - jar commands

9.1 What are jar files

jar file are used to bundle class files, metadata and resources into one file for distribution. They follow the zip format.

9.2 Create jar

```
spyros@spyros-ubuntu-23:~$ jar -cvf jarname.jar myfolderforjar
$ Above command will create a jar with name jarname.jar and
$ will put in it all files contained in directory myfolderforjar

$ Note: It will put files in same filepath as in myfolderforjar
$ within jar, directory structure is important

$ -c: create jar
$ -v: verbose output
$ -f: specify the archive file name
```

9.3 Extract files from jar

```
spyros@spyros-ubuntu-23:~$ jar -xvf jarname.jar
$ Above command will extract all files from jar

spyros@spyros-ubuntu-23:~$ jar -xvf jarname.jar file1.class file2.class
$ Above command will extract only given files from jar

$ -x: extract files
```

9.4 Add/Update file in jar

```
spyros@spyros-ubuntu-23:~$ jar -uvf jarname.jar myfilesforjar
$ Above command will create directories and add files that
$ don't exist and will update existing files

-u: update jar
```

9.5 Delete file from jar

```
$ jar command doesn't have an option to delete file/directory from jar

$ Method #1: Extract all files, delete required files from
$ extracted files and create again the jar

$ Method #2: As said jar files are built on zip format, so any zip
$ program can be used to delete the file from the jar

$ Example using the zip command
spyros@spyros-ubuntu-23:~$ zip -d jarname.jar filetodelete.class
$ Above command will delete filetodelete.class from jar jarname.jar

$ -d: delete file/directory (used with zip command)
```

9.6 Java Decompiler - JD-GUI

One convenient way to view the content of a jar in a Graphical User Interface is jd-gui. For more on jd-gui please click here (JD-GUI Official Site).
Note: JD-GUI allows you only to view jar contents, not to edit jar content in any way.

10 Popular Linux Distros

1. Ubuntu
2. Debian
3. OpenSuse
4. Fedora
5. ElementaryOS
6. Arch Linux
7. Gentoo

11 Pending Items

- sed, awk
- vi, vim
- nano, pico
- exec & execdir

- Bash Script Programming
- dos2unix