

REMEMBERING LINUX...



Reviewing what you need from **Operating Systems** for this course



JOSÉ MANUEL REDONDO LÓPEZ PROYECTO "S-81 ISAAC PERAL" v4.0



Departamento de Informática
Universidad de Oviedo



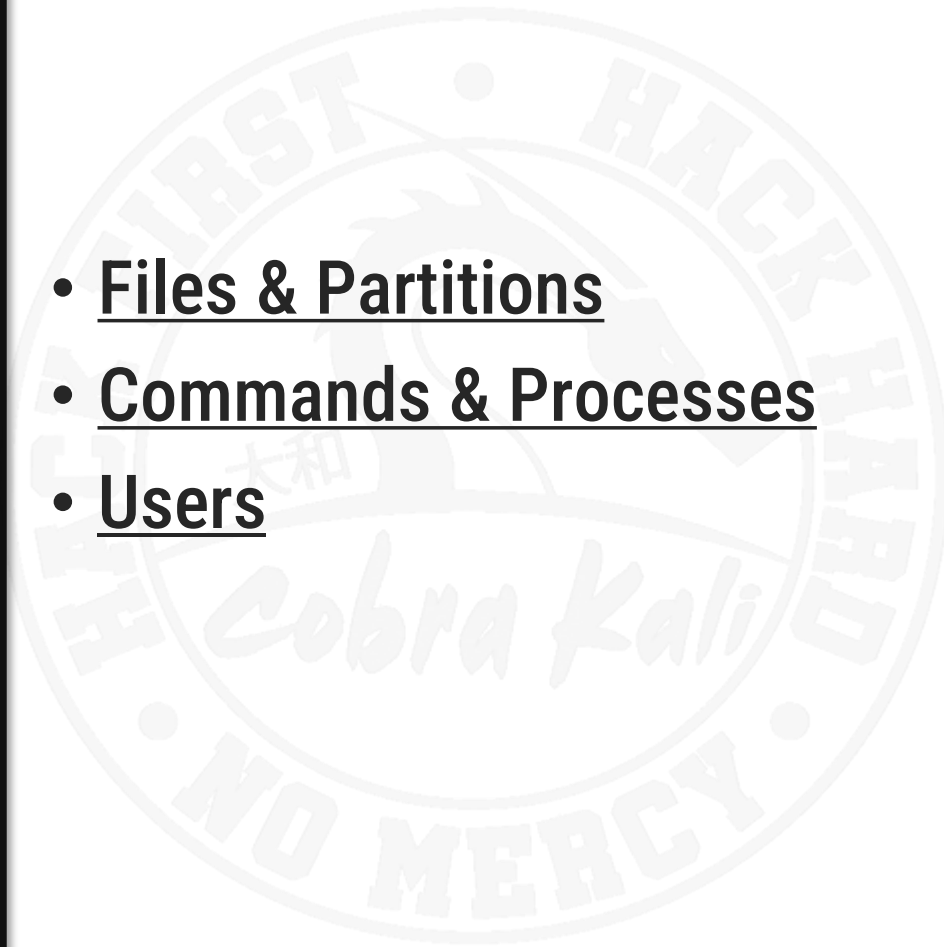
Universidad de Oviedo

Logo: @creative_vanesa (Instagram)



INDEX

- Files & Partitions
- Commands & Processes
- Users



WHAT IS THE PURPOSE OF THIS PRESENTATION?

- In this course there are several aspects to consider from the point of view of the inner workings of an OS
- To be able to handle them properly, you need to use concepts that you saw in the subject of **Operating Systems**
 - But the experience of these years has shown that not everyone remembers them
- Therefore, we have decided to make this small presentation to remind them to those who need it
- This content is **NOT EVALUABLE** in our course, it is only for help purposes

MACHINE ENTITIES FROM A SECURITY POINT OF VIEW



Computer Security

- Once inside a machine, there are three main entities we must consider from a security point of view
 - **Users**
 - **Files** and partitions
 - **Programs** (software)
- We will deal with them more thoroughly in the OS security topic
- But the entities itself were reviewed in the **SO** course

[< Go to Index](#)

FILES & PARTITIONS

What you can find on your hard drive



Departamento de Informática
Universidad de Oviedo



Universidad de Oviedo

FILES AND PARTITIONS



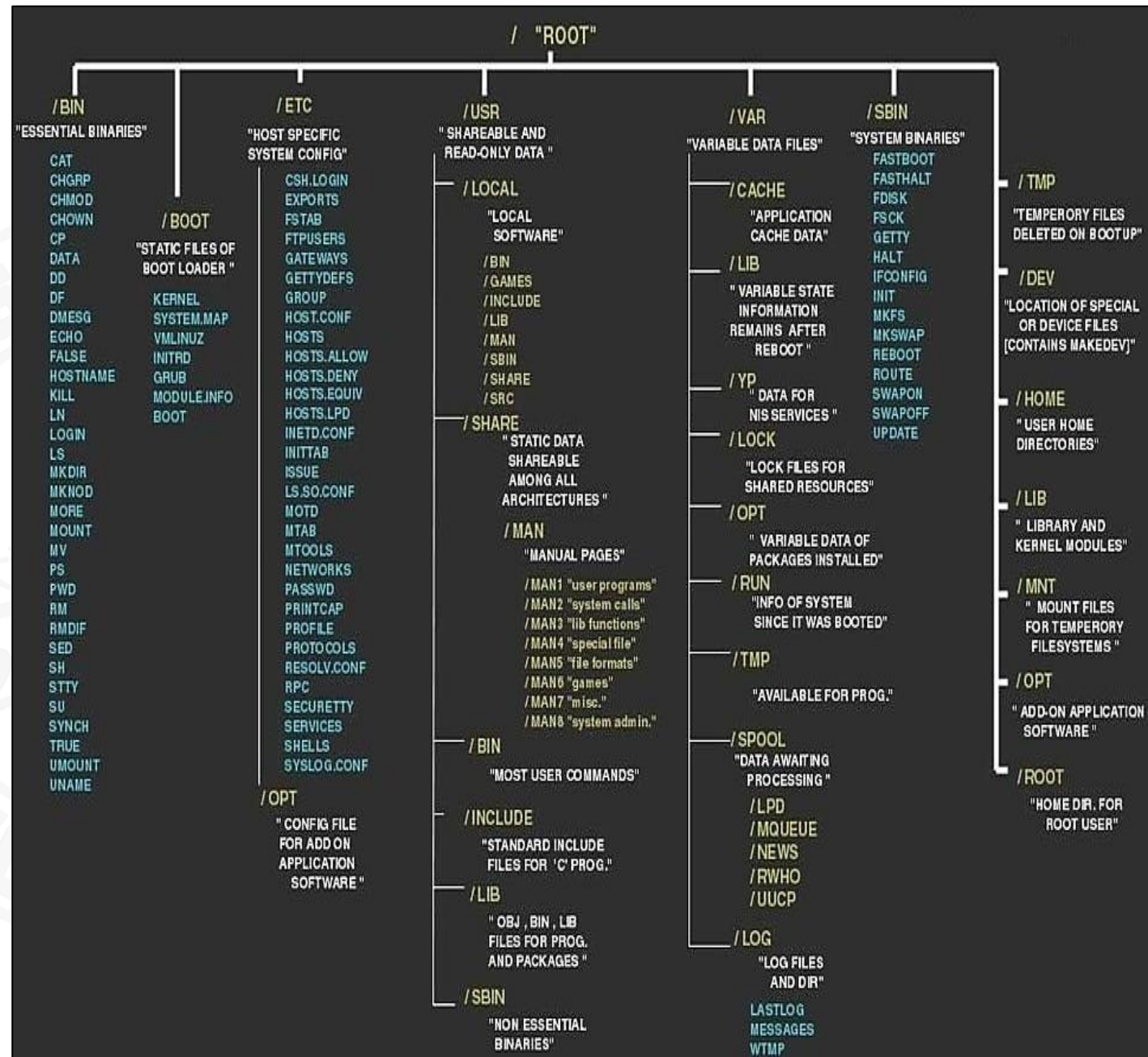
Source: <https://www.wifislax.com/sistema-de-archivos-linux-en-una-sola-imagen/>

- Every OS has a predefined structure of directories, files, and partitions

- Partitions group files under a directory in an isolated storage space
- Typically, transparent to the user

- The image shows a typical Linux installation

- We need to know where are the critical parts of the filesystem
 - Private files, device files, important services, logs...
- Without this knowledge, we cannot perform security operations effectively

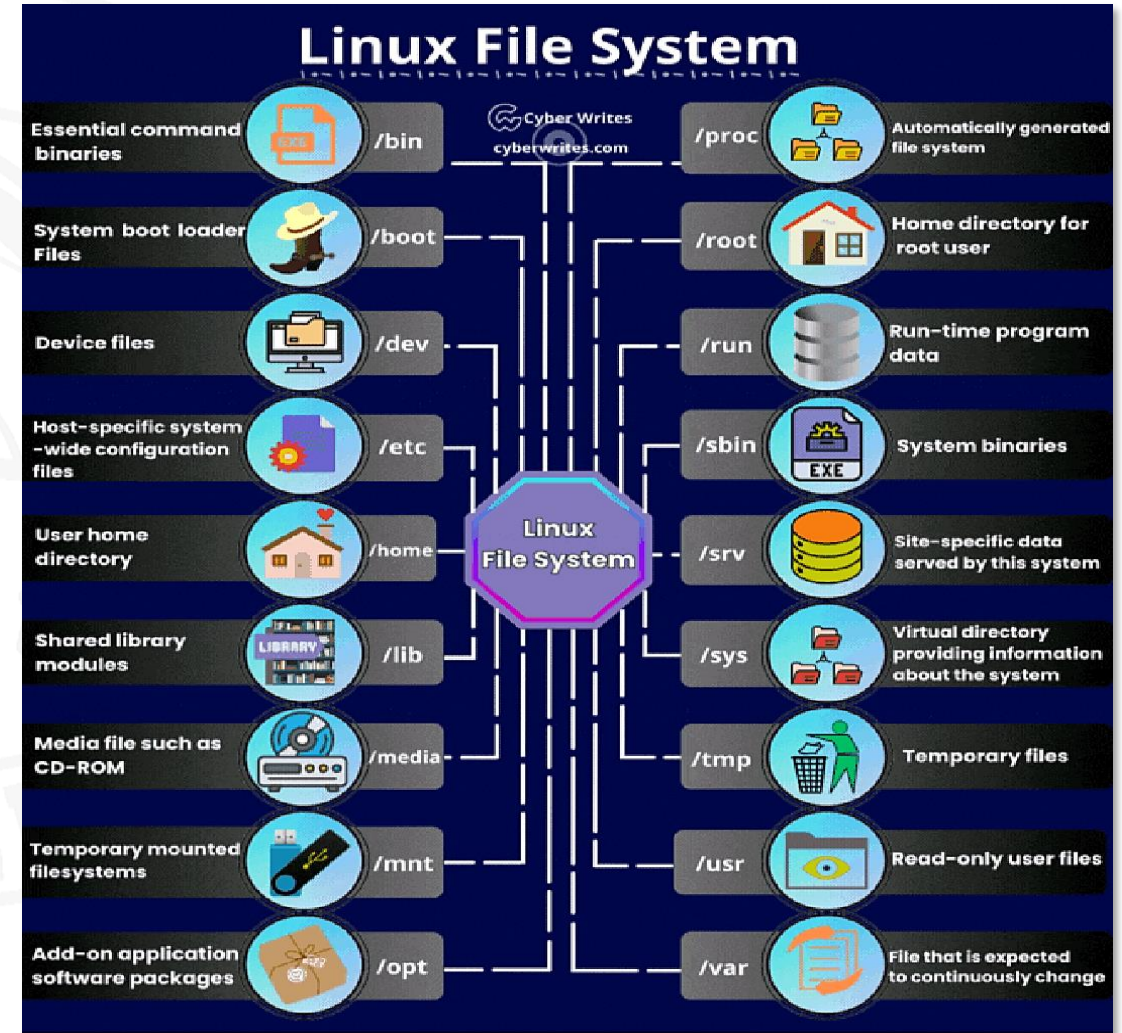
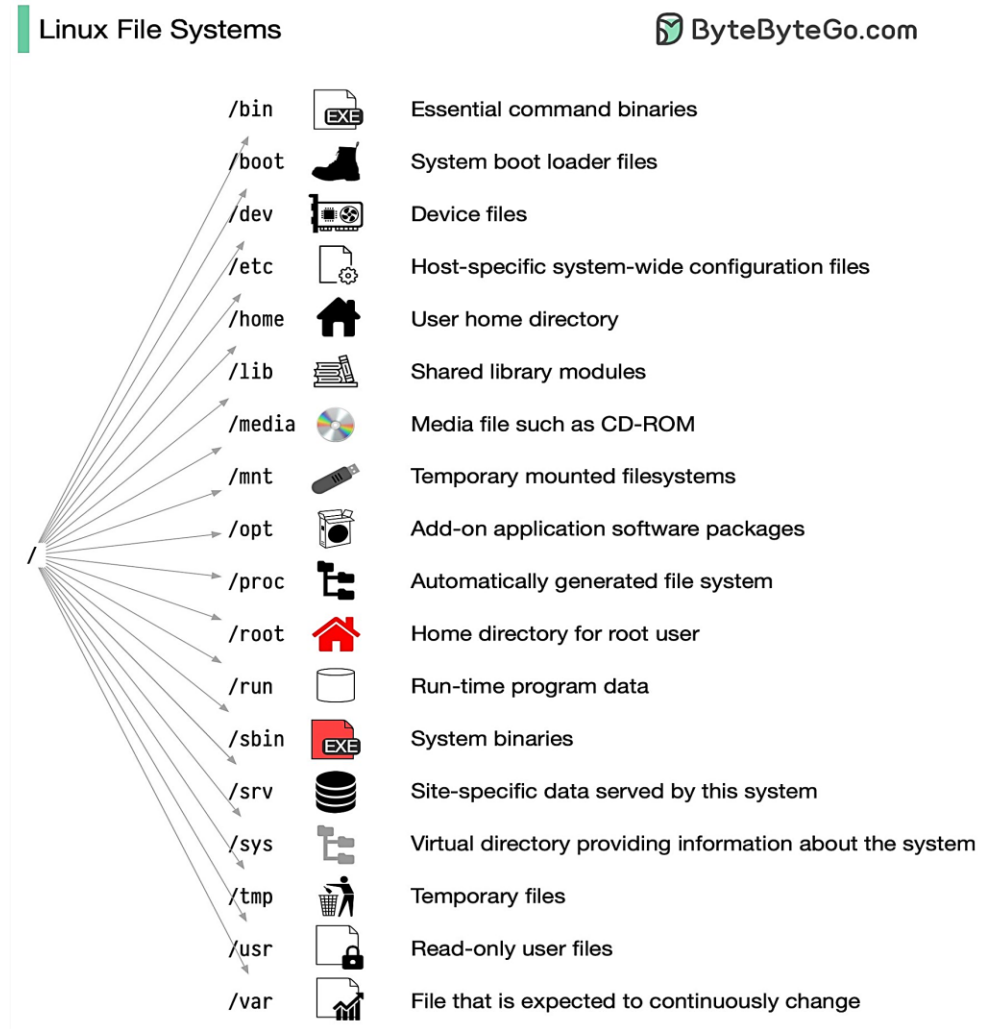


FILES & PARTITIONS



Computer Security

- If the previous diagram is too detailed, and you just want to know the purpose of each directory, you can use either of these two



FILE PERMISSIONS



- If you don't remember how they are managed, these two diagrams can surely help you

Linux File Permissions

blog.bytebytego.com

Binary	Octal	String Representation	Permissions
000	0 (0+0+0)	---	No Permission
001	1 (0+0+1)	--x	Execute
010	2 (0+2+0)	-w-	Write
011	3 (0+2+1)	-wx	Write + Execute
100	4 (4+0+0)	r--	Read
101	5 (4+0+1)	r-x	Read + Execute
110	6 (4+2+0)	rw-	Read + Write
111	7 (4+2+1)	rwX	Read + Write + Execute

Owner			Group			Other		
r	W	x	r	W	-	r	-	x
r	Read	4	r	Read	4	r	Read	4
w	Write or Edit	2	w	Write or Edit	2	-	No Permission	0
x	Execute	1	-	No Permission	0	x	Execute	1
7			6			5		

Linux file permissions cheatsheet

@sysxplore

```
sysxplore@zorinos:~$ ls -l
total 93792
```

```
-rwx-rw-r-- 1 1 sysxplore sysxplore 20 Jul 18 19:14 backup.sh
-rw-rw-r-- 1 2 traw traw 0 Jun 23 12:59 demo
drwxr-xr-x 2 4096 May 15 10:42 Desktop
```

Number of linked hard-links
Owner of the file
Group that own the file
File size in bytes
File modification date and time
File name

User Owner Permissions
Group Owner Permissions
Other Permissions

-rwxrw-r--

r	Read	4
w	Write	2
x	Execute	1
7		

r	Read	4
w	Write	2
-	No permission	0
6		

r	Read	4
-	No permission	0
-	No permission	0
4		

SUID Permission

-rwsrw-r--

\$ chmod u+s file

SGID Permission

drwxrwsr--

\$ chmod g+s directory_name

Stick Bit Permission

drwxrwxrwt

\$ chmod +t directory_name

Binary	Octal	Permissions	Representation
000	0 (0+0+0)	No Permission	---
001	1 (0+0+1)	Execute	--x
010	2 (0+2+0)	Write	-w-
011	3 (0+2+1)	Write + Execute	-wx
100	4 (4+0+0)	Read	r--
101	5 (4+0+1)	Read + Execute	r-x
110	6 (4+2+0)	Read + Write	rw-
111	7 (4+2+1)	Read + Write + Execute	rwX

Owner			Group			Other		
r	w	x	r	w	x	r	w	x
S			S			T		

Capital S is an error it occurs if you set SUID bit or SGID bit to a file without execute (x) bit set
Capital T is an error it occurs if you set Stick bit to a file without execute (x) bit set

@sysxplore

[< Go to Index](#)

COMMANDS & PROCESSES

Software you may need to use during the course



Departamento de Informática
Universidad de Oviedo



Universidad de Oviedo



● Commands: invoked by users to achieve different effects

- The image shows the typical commands of a Linux system
- More complete listing:

<https://ss64.com/bash/>

● Background running applications

- They are called **services** (Windows) or **Daemons** (Linux)
- Implement different features of the OS
 - Most are required to its proper operation
- They are running continuously or periodically (programmed)
 - Or once a certain application runs

File Commands

ls - directory listing
ls -al - formatted listing with hidden files
cd dir - change directory to *dir*
cd - change to home
pwd - show current directory
mkdir dir - create a directory *dir*
rm file - delete *file*
rm -r dir - delete directory *dir*
rm -f file - force remove *file*
rm -rf dir - force remove directory *dir* *
cp file1 file2 - copy *file1* to *file2*
cp -r dir1 dir2 - copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv file1 file2 - rename or move *file1* to *file2*
 if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s file link - create symbolic link *link* to *file*
touch file - create or update *file*
cat > file - places standard input into *file*
more file - output the contents of *file*
head file - output the first 10 lines of *file*
tail file - output the last 10 lines of *file*
tail -f file - output the contents of *file* as it grows, starting with the last 10 lines

Process Management

ps - display your currently active processes
top - display all running processes
kill pid - kill process id *pid*
killall proc - kill all processes named *proc* *
bg - lists stopped or background jobs; resume a stopped job in the background
fg - brings the most recent job to foreground
fg n - brings job *n* to the foreground

File Permissions

chmod octal file - change the permissions of *file* to *octal*, which can be found separately for user, group, and world by adding:

- 4 - read (r)
- 2 - write (w)
- 1 - execute (x)

Examples:

chmod 777 - read, write, execute for all
chmod 755 - rwx for owner, rx for group and world
 For more options, see **man chmod**.

SSH

ssh user@host - connect to *host* as *user*
ssh -p port user@host - connect to *host* on port *port* as *user*
ssh-copy-id user@host - add your key to *host* for *user* to enable a keyed or passwordless login

Searching

grep pattern files - search for *pattern* in *files*
grep -r pattern dir - search recursively for *pattern* in *dir*
command | grep pattern - search for *pattern* in the output of *command*
locate file - find all instances of *file*

System Info

date - show the current date and time
cal - show this month's calendar
uptime - show current uptime
w - display who is online
whoami - who you are logged in as
finger user - display information about *user*
uname -a - show kernel information
cat /proc/cpuinfo - cpu information
cat /proc/meminfo - memory information
man command - show the manual for *command*
df - show disk usage
du - show directory space usage
free - show memory and swap usage
whereis app - show possible locations of *app*
which app - show which *app* will be run by default

Compression

tar cf file.tar files - create a tar named *file.tar* containing *files*
tar xf file.tar - extract the files from *file.tar*
tar czf file.tar.gz files - create a tar with Gzip compression
tar xzf file.tar.gz - extract a tar using Gzip
tar cjf file.tar.bz2 - create a tar with Bzip2 compression
tar xjf file.tar.bz2 - extract a tar using Bzip2
gzip file - compresses *file* and renames it to *file.gz*
gzip -d file.gz - decompresses *file.gz* back to *file*

Network

ping host - ping *host* and output results
whois domain - get whois information for *domain*
dig domain - get DNS information for *domain*
dig -x host - reverse lookup *host*
wget file - download *file*
wget -c file - continue a stopped download

Installation

Install from source:

./configure
make
make install
dpkg -i pkg.deb - install a package (Debian)
rpm -Uvh pkg.rpm - install a package (RPM)

Shortcuts

Ctrl+C - halts the current command
Ctrl+Z - stops the current command, resume with **fg** in the foreground or **bg** in the background
Ctrl+D - log out of current session, similar to **exit**
Ctrl+W - erases one word in the current line
Ctrl+U - erases the whole line
Ctrl+R - type to bring up a recent command
!! - repeats the last command
exit - log out of current session

* use with extreme caution.



TYPICAL LINUX COMMANDS



- If the commands in the previous image are too many, here you can find the most common ones
- In the course you will have to use a small set of them
 - ;But it doesn't hurt to have a quick reference just in case! 😊

Source: <https://twitter.com/Sheraj99/status/1622735819298545664>

{ Linux Commands Cheat Sheet }

1 ls

● The most frequently used command in Linux to list directories

4 mkdir

● Command used to create directories in Linux

7 rm

● Delete files or directories

10 ca

● Display file contents on the terminal

13 less

● Linux command to display paged outputs in the terminal

16 whoami

● Get the active username

18 grep

● Search for a string within an output

21 diff

● Find the difference between two files

24 sort

● to sort the content of a file while outputting

27 unzip

● Unzip files in Linux

30 ps

● Display active processes

33 mount

● Mount file systems in Linux

36 ifconfig

● Display network interfaces and IP addresses

39 ufw

● UFW firewall command

42 sudo

● Command to escalate privileges in Linux

46 dd

● Majorly used for creating bootable USB sticks

46 whereis

● Locate the binary, source, and manual pages for a command

49 useradd and usermod

● Add new user or change existing users data

2 pwd

● Print working directory command in Linux

6 mv

● Move or rename files in Linux

8 touch

● Create blank/empty files

11 clear

● Clear the terminal display

14 man

● Access manual pages for all Linux commands

17 tar

● Command to extract and compress files in Linux

19 head

● Return the specified number of lines from the top

22 cmp

● Allows you to check if two files are identical

25 export

● Export environment variables in Linux

28 ssh

● Secure Shell command in Linux

31 kill and killall

● Kill active processes by process ID or name

34 chmod

● Command to change file permissions

37 traceroute

● Trace all the network hops to reach the destination

40 iptables

● Base firewall for all other firewall utilities to interface with

43 cal

● View a command-line calendar

47 whatis

● LFind what a command is used for

50.passwd

● pdate passwords

3 cd

● Linux command to navigate through directories

6 cp

● Similar usage as mv but for copying files in Linux

9 ln

● Create symbolic links (shortcuts) to other files

12 echo

● Print any text that follows the command

15 uname

● Linux command to get basic information

20 tail

● Return the specified number of lines from the bottom

23 comm

● Combines the functionality of diff and cmp

26 zip

● Zip files in Linux

29 service

● Linux command to start and stop services

32 df

● Display disk filesystem information

35 chown

● Command for granting ownership of files or folders

38 wget

● Direct download files from the internet

41 apt

● Package managers depending on the distro

44 alias

● Create custom shortcuts for your regularly used commands

48 top

● View active processes live with their system usage

SSH



- It is a tool that allows you to establish secure remote connections with servers
- Used to authenticate and encrypt connections
- Enables remote control of servers and secure data transfer



Linux OpenSSH command





```
ssh -p 3000 alice@hostIP
```

Connect to <hostIP> over SSH on custom SSH port 3000 (port 22 when omitted)



```
ssh -i /path/to/private-key alice@hostIP
```

Use SSH key authentication instead of password authentication



```
ssh alice@hostIP "ls -al"
```

Execute a non-interactive command (e.g. `ls -al`) on a remote host directly over SSH



```
ssh -L <local-port>:<hostIP1>:<remote-port> alice@hostIP2
```

Forward traffic at <local-port> to <hostIP1>:<remote-port> via jump server <hostIP2>



```
ssh -J bob@hostIP1:port1 alice@hostIP2
```

Connect to <hostIP2> via an intermediate jump server running at <hostIP1>:<port1>



```
ssh-copy-id -i /path/to/public-key alice@hostIP
```

Copy the user's public key to <hostIP> for SSH key authentication



```
ssh -F /path/to/ssh_config alice@hostIP
```

Connect to <hostIP> with custom options defined in SSH config file



```
ssh -fN -R <remote-port>:localhost:22 alice@hostIP
```

Forward traffic on port <remote-port> of <hostIP> to port 22 of local server



```
scp -rP 3000 /path/to/local-dir alice@hostIP:/path/to/remote-dir
```

Transfer local directory to remote path hosted at <hostIP> on custom SSH port 3000



This post is originally created by @dan_nanni on Instagram under CC BY-SA 3.0 license. You are free to redistribute it as long as you keep this notice.

CHEAT SHEET

SSH - common commands and secure config

BlowStack

SSH connections

```
connects to a server (default port 22)
$ ssh user@server

uses a specific port declared in sshd_config
$ ssh user@server -p other_port

runs a script on a remote server
$ ssh user@server script_to_run

compresses and downloads from a remote server
$ ssh user@server "tar cvzf - ~/source" > output.tgz

specifies other ssh key for connection
$ ssh -i ~/.ssh/specific_ssh_fkey
```

SSH service

```
starts ssh service
$ (sudo) service ssh start

checks ssh service status
$ (sudo) service ssh status

stops ssh service
$ (sudo) service ssh stop

restarts ssh service
$ (sudo) service ssh restart
```

SCP (Secure Copy)

```
copies a file from a remote server to a local machine
$ scp user@server:/directory/file.ext local_destination/

copies a file between two servers
$ scp user@server:/dir/file.ext user@server:/dir

copies a file from a local machine to a remote server
$ scp local_destination/file.ext user@server:/directory

uses a specific port declared for SSH in sshd_config
$ scp -P port

copies recursive a whole folder
$ scp -r user@server:/directory local_destination/

copies all files from a folder
$ scp user@server:/directory/* local_destination/

copies all files from a server folder to the current folder
$ scp user@server:/directory/* .

compresses data on network using gzip
$ scp -C

prints verbose info about the current transfer
$ scp -v
```

Full articles about cyber security at
<https://blowstack.com/blog/cyber-security>

SSH keys

```
generates a new ssh key
$ ssh-keygen -t rsa -b 4096

sends the key to the server
$ ssh-copy-id user@server

converts ids_rsa into ppk
$ puttygen current_key -o keyname.ppk
```

SSH config

```
opens config file (usual location)
$ sudo nano /etc/ssh/sshd_config

changes default SSH port (22)
Port 9809

disables root login
PermitRootLogin no

restricts access to specific users
AllowUsers user1, user2

enables login through ssh key
PubkeyAuthentication yes

disables login through password
PasswordAuthentication no

disables usage of files .rhosts and .shosts
IgnoreRhosts yes

disables a less secure type of login
HostbasedAuthentication no

number of unauthenticated connections before dropping
MaxStartups 10:30:100

no. of failed tries before the servers stops accepting new tries
MaxAuthTries 3

max current ssh sessions
MaxSessions 1

disables interactive password authentication
ChallengeResponseAuthentication no

no empty password allowed
PermitEmptyPasswords no

disables Rhost authentication
RhostsAuthentication no

disables port forwarding (blocks i.e MySQL Workbench)
AllowTcpForwarding no
X11Forwarding no

prints much more info about SSH connections
LogLevel VERBOSE
```

- **Command that allows you to download files from the Internet**
 - Supports downloads via FTP, SFTP, HTTP and HTTPS
- **It's useful for downloading content, from individual files to entire webs**





The wget command downloads files from the Internet.

Downloads	
<code>wget http://example.com</code>	Download the index page of example.com
<code>wget http://example.com \</code> <code>--output-document foo.html</code>	Save index.html as foo.html
<code>wget http://example.com \</code> <code>--output-document -</code>	Display output in terminal
<code>wget --continue</code> <code>https://example.com/file.iso</code>	Resume a partial download
<code>wget example.com/A{1..12}.jpg</code>	Download A1.jpg , A2.jpg , up to A12.jpg
<code>wget --mirror example.com</code>	Download the entire example.com site

Headers	
<code>--debug</code>	View HTML headers
<code>--headers="User:Agent: mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124"</code>	Alter your User Agent to appear as the Chrome browser running on Windows 10

Redirect	
<code>--max-redirect 0</code>	Do not follow 301 redirects, but resolve URL

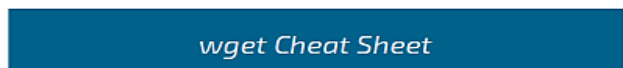
Seth Kenlon

CC BY-SA 4.0

Opensource.com


Supported by
Red Hat





Options	
<code>--base http://foo</code>	Substitute relative paths (such as . and ..) in downloaded HTML with the URL http://foo
<code>--tries inf</code>	Try all but 404 downloads infinite times until successfully downloaded. Default: 20
<code>--no-clobber</code>	Do not save over an existing downloaded file
<code>--backups 12</code>	If a file exists locally, append a number (up to 12) to its file name to create a backup.
<code>--progress=dot</code>	Show download progress with dots on screen
<code>--timestamping</code>	Set downloaded file timestamps to match those of the remote server
<code>--no-use-server-timestamps</code>	Do not use server timestamps
<code>--spider</code>	Do not download files, just crawl the site
<code>--timeout 30</code>	Stop an unsuccessful download attempt after 30 seconds. Default: 900 seconds
<code>--rate-limit 60k</code>	Limit download rate to 60 KB per second
<code>--wait 6</code>	Wait 6 seconds between downloads
<code>--random-wait</code>	Multiply --wait value by a number from 0.5 to 1.5
<code>--user tux --password foo</code>	Send user name tux and password foo
<code>--ask-password</code>	Prompt for interactive password

Seth Kenlon

CC BY-SA 4.0

Opensource.com


Supported by
Red Hat

CURL



- Tool to transfer data to or from a server using different protocols such as HTTP, FTP, POP3, among others
- It's useful for verifying connectivity to URLs and transferring data



hide progress	verbose	extra info	output	timeout
-s	-v --trace-ascii <file>	-w "format"	-O -o <file>	-m <seconds>
POST	POST encoded	multipart formpost	PUT	HEAD (ers too)
-d "string" -d @file	--data-urlencode "[name]=val"	-F name=value -F name=@file	-T <file>	-I -i
custom method	read cookiejar	write cookiejar	send cookies	user-agent
-X "METHOD"	-b <file>	-c <file>	-b "c=1; d=2"	-A "string"
proxy	add/remove headers	custom address	smaller data	insecure HTTPS
-x <host:port>	-H "name: value" -H "name:"	--resolve <host:port:addr>	--compressed	-k
Basic auth	follow redirects	parallel	generate code	list options
-u user:passwd	-L	-Z	--libcurl <file>	--help

Fuente:

<https://daniel.haxx.se/blog/2020/01/20/curl-cheat-sheet-refresh/>

[< Go to Index](#)

USERS

The ones who can actually do something in the OS



Departamento de Informática
Universidad de Oviedo



Universidad de Oviedo

USERS ACCORDING TO THEIR CAPABILITIES: PRIVILEGED USER ACCOUNTS

- These have more privileges than “ordinary” users
 - Able to install/remove software, upgrade the OS, modify system or application configurations...
- They might also have access to files that are not normally accessible to standard users
- Normally we have these privileged accounts
 - **Root / Administrator account:** the system administrator
 - Able to perform any change in the OS <https://www.ssh.com/iam/user/root/>
 - **System accounts:** usually with known user IDs
 - Created during the OS installation for management
 - **Service accounts:** run applications/other processes
 - Also created to own data and configuration files
 - They are not intended to be allowed interactive login (bash terminals)
 - Or be used by humans in any way
 - Only for administrative operations

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/
n
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/no
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
```



USERS ACCORDING TO THEIR CAPABILITIES: “ORDINARY” USER ACCOUNTS

● OS users that only perform “normal” operations

- Like creating documents, editing photos, browsing, playing games... any operation that **do not require any special privilege**
- Some of them may be **sudoers**
 - Tools like `sudo` can be used to grant certain users **the temporal ability to run selected commands as root**
 - This way, it is not necessary to enable a `root` account with interactive login capabilities
 - Only selected users (the ones in the `sudoers` group) can temporarily “promote” themselves to `root` to execute administrative operations
 - Once the operation ends, the user returns to their normal privilege status
 - Not every user is a `sudoer`
 - Any beyond the first created user (that is usually a sudoer automatically) must be manually added to this group

● In case of attack, the criminal will try to log in as `root` or as a sudoer

- If it is not possible, it will try to perform a **privilege scalation**
- We will deal with this later in the year!

REMEMBERING LINUX...

