# The Ultimate Linux Administration Cheat Sheet

by Andrei Dumitrescu

## TABLE OF CONTENTS

# The Linux Terminal

---

### Getting Help in Linux

**MAN Pages**
man command  # => Example: man ls

The man pages are displayed by the **less** command.

**SHORTCUTS:**
h  => getting help
q  => quit
enter  => show next line
space  => show next screen
/string  => search forward for a string
?string  => search backwards for a string
n / N  => next/previous appearance

```
# checking if a command is a shell built-in or an executable file
type rm          # => rm is /usr/bin/rm
type cd          # => cd is a shell builtin

# getting help for shell built-in commands
help command     # => Example: help cd
command --help   # => Example: rm --help

# searching for a command, feature or keyword in all man pages
```

man -k uname
man -k "copy files"
apropos passwd

## Keyboard Shortcuts

# autocompletes the command or the filename if its unique
TAB

# displaying all commands or filenames that start with written letters
TAB TAB (press twice)

# clearing the current line
CTRL + L

# closing the shell (exit)
CTRL + D

# cutting (removing) the current line
CTRL + U

# moving the cursor to start of the line
CTRL + A

# moving the cursor to the end of the line
Ctrl + E

# stopping the current command
CTRL + C

# sleeping a the running program
CTRL + Z

# opening a terminal
CTRL + ALT + T

## Bash History

# displaying the history
history

# removing a line (example: 100) from the history
history -d 100

# removing the entire history

```
history -c

# printing the number of commands saved in the history file (~/.bash_history)
echo $HISTFILESIZE

# printing the number of history commands saved in the memory
echo $HISTSIZE

# rerunning the last command from the history
!!

# running    a specific command from the history (example: the 20th command)
!20

# running the last nth (example: 10th) command from the history
!-10

# running the last command starting with abc
!abc

# printing the last command starting with abc
!abc:p

# reverse searching into the history
CTRL + R

# recording the date and time of each command in the history
HISTTIMEFORMAT="%d/%m/%y %T "    # => write it in ~/.bashrc to make it persistent
```

## Getting root access (sudo, su)

---

```
# running a command as root
# available only to the users that belong to sudo group [Ubuntu] or wheel [CentOS]
sudo command

# becoming root temporarily in the terminal
sudo su          # => enter the user's password

# setting the root password
sudo passwd root

# changing a user's password
passwd username
```

```
# becoming root temporarily in the terminal (available only if root has a password set)
su        # => enter the root password
```

## Linux Paths

---

**Paths:**
- absolute
- relative

**Any absolute path starts with /**

```
.           # => the current working directory
..          # => the parent directory
~           # => the user's home directory

cd          # => changing the current directory to user's home directory
cd ~        # => changing the current directory to user's home directory
cd -        # => changing the current directory to the last directory
cd /path_to_dir         # => changing the current directory to path_to_dir
pwd                     # => printing the current working directory

# installing tree
sudo apt install tree

tree directory/         # => Example: tree .
tree -d .               # => print only directories
tree -f .               # => print absolute paths
```

## The ls Command

---

```
ls [OPTIONS] [FILES]
# ~ => user's home directory
# . => current directory
# .. => parent directory

# listing the current directory
ls
ls .

# listing more directories
ls ~ /var /
```

# -l => long listing
ls -l ~

# -a => listing all files and directories including hidden ones
ls -la ~

# -1 => listing on a single column
ls -1 /etc

# -d => displaying information about the directory, not about its contents
ls -ld /etc

# -h => displaying the size in human readable format
ls -h /etc

# -S => displaying sorting by size
ls -Sh /var/log

# Note: ls does not display the size of a directory and all its contents. Use du instead
du -sh ~

# -X => displaying sorting by extension
ls -lX /etc

# --hide => hiding some files
ls --hide=*.log /var/log

# -R => displaying a directory recursively
ls -lR ~

# -i => displaying the inode number
ls -li /etc

## File Timestamps and Date

---

# displaying atime
ls -lu

# displaying mtime
ls -l
ls -lt

# displaying ctime

```
ls -lc

# displaying all timestamps
stat file.txt

# displaying the full timestamp
ls -l --full-time /etc/

# creating an empty file if it does not exist, update the timestamps if the file exists
touch file.txt

# changing only the access time to current time
touch -a file

# changing only the modification time to current time
touch -m file

# changing the modification time to a specific date and time
touch -m -t 201812301530.45 a.txt

# changing both atime and mtime to a specific date and time
touch -d "2010-10-31 15:45:30" a.txt

# changing the timestamp of a.txt to those of b.txt
touch a.txt -r b.txt

# displaying the date and time
date

# showing this month's calendar
cal

# showing the calendar of a specific year
cal 2021

# showing the calendar of a specific month and year
cal 7 2021

# showing the calendar of previous, current and next month
cal -3

# setting the date and time
date --set="2 OCT 2020 18:00:00"

# displaying the modification time and sorting the output by name.
```

```
ls -l

# displaying the output sorted by modification time, newest files first
ls -lt

# displaying and sorting by atime
ls -ltu

# reversing the sorting order
ls -ltu --reverse
```

## Viewing files (cat, less, more, head, tail, watch)

---

```
# displaying the contents of a file
cat filename

# displaying more files
cat filename1 filename2

# displaying the line numbers
can -n filename

# concatenating 2 files
cat filename1 filename2 > filename3

# viewing a file using less
less filename
```

**less shortcuts:**

```
h          => getting help
q          => quit
enter      => show next line
space       => show next screen
/string    => search forward for a string
?string    => search backwards for a string
n / N      => next/previous appearance

# showing the last 10 lines of a file
tail filename

# showing the last 15 lines of a file
tail -n 15 filename
```

# showing the last lines of a file starting with line number 15
tail -n +5 filename

# showing the last 10 lines of the file in real-time
tail -f filename

# showing the first 10 lines of a file
head filename

# showing the first 15 lines of a file
head -n 15 filename

# running repeatedly a command with refresh of 3 seconds
watch -n 3 ls -l

## Working with files and directory (touch, mkdir, cp, mv, rm, shred)

---

# creating a new file or updating the timestamps if the file already exists
touch filename

# creating a new directory
mkdir dir1

# creating a directory and its parents as well
mkdir -p mydir1/mydir2/mydir3

### The cp command

# copying file1 to file2 in the current directory
cp file1 file2

# copying file1 to dir1 as another name (file2)
cp file1 dir1/file2

# copying a file prompting the user if it overwrites the destination
cp -i file1 file2

# preserving the file permissions, group and ownership when copying
cp -p file1 file2

# being verbose
cp -v file1 file2

# recursively copying dir1 to dir2 in the current directory
cp -r dir1 dir2/

# copy more source files and directories to a destination directory
cp -r file1 file2 dir1 dir2 destination_directory/

## The mv command

# renaming file1 to file2
mv file1 file2

# moving file1 to dir1
mv file1 dir1/

# moving a file prompting the user if it overwrites the destination file
mv -i file1 dir1/

# preventing a existing file from being overwritten
mv -n file1 dir1/

# moving only if the source file is newer than the destination file or when the destination file is missing
mv -u file1 dir1/

# moving file1 to dir1 as file2
mv file1 dir1/file2

# moving more source files and directories to a destination directory
mv file1 file2 dir1/ dir2/ destination_directory/

## The rm command

# removing a file
rm file1

# being verbose when removing a file
rm -v file1

# removing a directory
rm -r dir1/

```
# removing a directory without prompting
rm -rf dir1/

# removing a file and a directory prompting the user for confirmation
rm -ri fil1 dir1/

# secure removal of a file (verbose with 100 rounds of overwriting)
shred -vu -n 100 file1
```

## Piping and Command Redirection

---

**Piping Examples:**

```
ls -lSh /etc/ | head                  # see the first 10 files by size
ps -ef | grep sshd                    # checking if sshd is running
ps aux --sort=-%mem | head -n 3    # showing the first 3 process by memory consumption
```

**Command Redirection**

```
# output redirection
ps aux > running_processes.txt
who -H > loggedin_users.txt

# appending to a file
id >> loggedin_users.txt

# output and error redirection
tail -n 10 /var/log/*.log > output.txt 2> errors.txt

# redirecting both the output and errors to the same file
tail -n 2 /etc/passwd /etc/shadow > output_errors.txt 2>&1

cat -n /var/log/auth.log | grep -ai "authentication failure" | wc -l
cat -n /var/log/auth.log | grep -ai "authentication failure" > auth.txt        # => piping and redirection
```

## Finding Files (find, locate)

---

## locate

```
# installing plocate
sudo apt install plocate

# updating the locate db
sudo updatedb

# displaying statistics
locate -S

# finding file by name
locate filename          # => filename is expanded to *filename*
locate -i filename       # => the filename is case insensitive
locate -b '\filename'    # => finding by exact name

# finding using the basename
locate -b filename

# finding using regular expressions
locate -r 'regex'

# checking that the file exists
locate -e filename

# showing command path
which command
which -a command
```

## find

```
find PATH OPTIONS
```

**Example:** `find ~ -type f -size +1M` `# => finding all files in ~ bigger than 1 MB`

**Options:**

```
 -type f, d, l, s, p
 -name filename
-iname filename     # => case-insensitive
 -size n, +n, -n
 -perm permissions
 -links n, +n, -n
 -atime n, -mtime n, ctime n
```

-user owner
-group group_owner

## Searching for text patterns (grep)

---

**grep [OPTIONS] pattern file**

**Options:**
-n # => print line number
-i # => case insensitive
-v # inverse the match
-w # search for whole words
-a # search in binary files
-R # search in directory recursively
-c # display only the number of matches
-C n # display a context (n lines before and after the match)

# printing ASCII chars from a binary file
strings binary_file # => Example: strings /bin/ls

## VIM

---

**Modes of operation:** Command, Insert, and Last Line Modes.
**VIM Config File:** ~/.vimrc

Entering the Insert Mode from the Command Mode
i => insert before the cursor
I => insert at the beginning of the line
a => insert after the cursor
A => insert at the end of the line
o => insert on the next line

Entering the Last Line Mode from the Command Mode
:

Returning to Command Mode from Insert or Last Line Mode
ESC

**Shortcuts in Last Line Mode:**

w!  => write/save the file
q!  => quit the file without saving
wq!  => save/write and quit
e!  => undo to the last saved version of the file
set no  => set line numbers
set nonu  => unset line numbers
syntax on|off
%s/search_string/replace_string/g

**Shortcuts in Command Mode:**
x  => remove char under the cursor
dd  => cut the current line
5dd  => cut 5 lines
ZZ  => save and quit
u  => undo
G  => move to the end of file
$  => move to the end of line
0 or ^  => move to the beginning of file
:n (Ex :10)  => move to line n
Shift+v  => select the current line
y  => yank/copy to clipboard
p  => paste after the cursor
P  => paste before the cursor
/string  => search for string forward
?string  => search for string backward
n  => next occurrence
N  => previous occurrence

# opening more files in stacked windows
vim -o file1 file2

# opening more files and highlighting the differences
vim -d file1 file2
Ctrl+w => move between files

# Account Management

---

**Important files:**
/etc/passwd  # => users and info: *username:x:uid:gid:comment:home_directory:login_shell*
/etc/shadow  # => users' passwords
/etc/group  # => groups

# creating a user account
useradd [OPTIONS] username

**Options:**
-m => create home directory
-d directory => specify another home directory
-c "comment"
-s shell
-G => specify the secondary groups (must exist)
-g => specify the primary group (must exist)

**Exemple:**
useradd -m -d /home/john -c "C++ Developer" -s /bin/bash -G sudo,adm,mail john

# changing a user account
usermod [OPTIONS] username # => uses the same options as useradd

**Example:**
usermod -aG developers,managers john # => adding the user to two secondary groups

# deleting a user account
userdel -r username # => -r removes user's home directory as well

# creating a group
groupadd group_name

# deleting a group
groupdel group_name

# displaying all groups
cat /etc/groups

# displaying the groups a user belongs to
groups

# creating admin users
# add the user to *sudo* group in Ubuntu and *wheel* group in CentOS
usermod -aG sudo john

## Monitoring Users

who -H # => displays logged in users
id # => displays the current user and its groups
whoami # => displays EUID

# listing who's logged in and what's their current process.
w
uptime

# printing information about the logins and logouts of the users
last
last -u username

## File Permissions

---

**Legend:**
u = user
g = group
o = others/world
a = all

r = read
w = write
x = execute
- = no access

# displaying the permissions (ls and stat)
ls -l /etc/passwd
    -rw-r--r-- 1 root root 2871 aug 22 14:43 /etc/passwd

stat /etc/shadow
    File: /etc/shadow
    Size: 1721  Blocks: 8  IO Block: 4096  regular file
    Device: 805h/2053d Inode: 524451  Links: 1
    Access: (0640/-rw-r-----)  Uid: (  0/  root)  Gid: (  42/  shadow)
    Access: 2020-08-24 11:31:49.506277118 +0300
    Modify: 2020-08-22 14:43:36.326651384 +0300
    Change: 2020-08-22 14:43:36.342652202 +0300
    Birth: -

# changing the permissions using the relative (symbolic) mode
chmod u+r filename
chmod u+r,g-wx,o-rwx filename
chmod ug+rwx,o-wx filename
chmod ugo+x filename
chmod a+r,a-wx filename

# changing the permissions using the absolute (octal) mode

| PERMISSIONS | | | EXAMPLE |
|---|---|---|---|
| u | g | o | |
| rwx | rwx | rwx | chmod 777 filename |
| rwx | rwx | r-x | chmod 775 filename |
| rwx | r-x | r-x | chmod 755 filename |
| rwx | r-x | --- | chmod 750 filename |
| rw- | rw- | r-- | chmod 664 filename |
| rw- | r-- | r-- | chmod 644 filename |
| rw- | r-- | --- | chmod 640 filename |

# setting the permissions as of a reference file
chmod --reference=file1 file2

# changing permissions recursively
chmod -R u+rw,o-rwx filename

## SUID (Set User ID)

# displaying the SUID permission
ls -l /usr/bin/umount
    -rw**s**r-xr-x 1 root root 39144 apr  2 18:29 /usr/bin/umount

stat /usr/bin/umount
    File: /usr/bin/umount
    Size: 39144  Blocks: 80  IO Block: 4096  regular file
    Device: 805h/2053d Inode: 918756  Links: 1
    Access: (4755/-rwsr-xr-x)  Uid: (  0/  root)  Gid: (  0/  root)
    Access: 2020-08-22 14:35:46.763999798 +0300
    Modify: 2020-04-02 18:29:40.000000000 +0300
    Change: 2020-06-30 18:27:32.851134521 +0300
    Birth: -

# setting SUID
chmod u+s executable_file
chmod 4XXX executable_file  # => **Example:** chmod 4755 script.sh

## SGID (Set Group ID)

# displaying the SGID permission
ls -ld projects/
    drwxr-**s**--- 2 student student 4096 aug 25 11:02 projects/

```
stat projects/
  File: projects/
  Size: 4096        Blocks: 8          IO Block: 4096   directory
  Device: 805h/2053d Inode: 266193      Links: 2
  Access: (2750/drwxr-s---)  Uid: ( 1001/ student)   Gid: ( 1002/ student)
  Access: 2020-08-25 11:02:15.013355559 +0300
  Modify: 2020-08-25 11:02:15.013355559 +0300
  Change: 2020-08-25 11:02:19.157290764 +0300
  Birth: -

# setting SGID
chmod 2750 projects/
chmod g+s projects/
```

## The Sticky Bit

```
# displaying the sticky bit permission
ls -ld /tmp/
  drwxrwxrwt 20 root root 4096 aug 25 10:49 /tmp/

stat /tmp/
  File: /tmp/
  Size: 4096        Blocks: 8          IO Block: 4096   directory
  Device: 805h/2053d Inode: 786434      Links: 20
  Access: (1777/drwxrwxrwt)  Uid: (    0/    root)   Gid: (    0/    root)
  Access: 2020-08-22 14:46:03.259455125 +0300
  Modify: 2020-08-25 10:49:53.756211470 +0300
  Change: 2020-08-25 10:49:53.756211470 +0300
  Birth: -

# setting the sticky bit
mkdir temp
chmod 1777 temp/
chmod o+t temp/
ls -ld temp/
  drwxrwxrwt 2 student student 4096 aug 25 11:04 temp/
```

## UMASK

```
# displaying the UMASK
umask
```

```
# setting a new umask value
umask new_value       # => Example: umask 0022
```

### Changing File Ownership (root only)

```
# changing the owner
chown new_owner file/directory        # => Example: sudo chown john a.txt

# changing the group owner
chgrp new_group file/directory

# changing both the owner and the group owner
chown new_owner:new_group file/directory

# changing recursively the owner or the group owner
chown -R new-owner file/directory

# displaying the file attributes
lsattr filename

#changing the file attributes
chattr +-attribute filename       # => Example: sudo chattr +i report.txt
```

## Processes

---

### Process Viewing (ps, pstree, pgrep)

```
# checking if a command is shell built-in or executable file
type rm          # => rm is /usr/bin/rm
type cd          # => cd is a shell builtin

# displaying all processes started in the current terminal
ps

# displaying all processes running in the system
ps -ef
ps aux
ps aux | less          # => piping to less

# sorting by memory and piping to less
ps aux --sort=%mem | less

# ASCII art process tree
ps -ef --forest
```

```
# displaying all processes of a specific user
ps -f -u username

# checking if a process called sshd is running
pgrep -l sshd           # matches against the process name
pgrep -f sshd           # matches against the full command line
ps -ef | grep sshd

#displaying a hierarchical tree structure of all running processes
pstree

# merging identical branches
pstree -c
```

## Dynamic Real-Time View of Processes(top)

```
# starting top
top
```

**top shortcuts while it's running**

```
h               # => getting help
space           # => manual refresh
d               # => setting the refresh delay in seconds
q               # => quitting top
u               # => display processes of a user
m               # => changing the display for the memory
1               # => individual statistics for each CPU
x/y             # => highlighting the running process and the sorting column
b               # => toggle between bold and text highlighting
<               # => move the sorting column to the left
>               # => move the sorting column to the right
F               # => entering the Field Management screen
W               # => saving top settings

# running top in batch mode (3 refreshes, 1 second delay)
top -d 1 -n 3 -b > top_processes.txt

# Interactive process viewer (top alternative)
sudo apt update && sudo apt install htop      # => Installing htop
htop
```

### Killing processes (kill, pkill, killall)

```
# listing all signals
kill -l

# sending a signal (default SIGTERM - 15) to a process by pid
kill pid            # => Example: kill 12547

# sending a signal to more processes
kill -SIGNAL pid1 pid2 pid3 ...

# sending a specific signal (SIGHUP - 2) to a process by pid
kill -2 pid
kill -HUP pid
kill -SIGHUP pid

# sending a signal (default SIGTERM - 15) to process by process name
pkill process_name              # => Example: pkill sleep
killall process_name
kill $(pidof process_name)   # => Example: kill -HUP $(pidof sshd)

# running a process in the background
command &     # => Example: sleep 100 &

# Showing running jobs
jobs

# Stopping (pausing) the running process
Ctrl + Z

# resuming and bringing to the foreground a process by job_d
fg %job_id

# resuming in the background a process by job_d
bg %job_id

# starting a process immune to SIGHUP
nohup command &       # => Example: nohup wget http://site.com &
```

# Networking

---

## Getting info about the network interfaces (ifconfig, ip, route)

```
# displaying information about enabled interfaces
ifconfig

# displaying information about all interfaces (enabled and disabled)
ifconfig -a
ip address show

# displaying info about a specific interface
ifconfig enp0s3
ip addr show dev enp0s3

# showing only IPv4 info
ip -4 address

# showing only IPv4 info
ip -6 address

# displaying L2 info (including the MAC address)
ip link show
ip link show dev enp0s3

# displaying the default gateway
route
route -n      # numerical addresses
ip route show

# displaying the DNS servers
systemd-resolve --status
```

## Setting the network interfaces (ifconfig, ip, route)

```
# disabling an interface
ifconfig enp0s3 down
ip link set enp0s3 down

# activating an interface
ifconfig enp0s3 up
ip link set enp0s3 up

# checking its status
ifconfig -a
ip link show dev enp0s3
```

# setting an ip address on an interface
ifconfig enp0s3 192.168.0.222/24 up
ip address del 192.168.0.111/24 dev enp0s3
ip address add 192.168.0.112/24 dev enp0s3

# setting a secondary ip address on sub-interface
ifconfig enp0s3:1 10.0.0.1/24

# deleting and setting a new default gateway
route del default gw 192.168.0.1
route add default gw 192.168.0.2

# deleting and setting a new default gateway
ip route del default
ip route add default via 192.168.0.1

# changing the MAC address
ifconfig enp0s3 down
ifconfig enp0s3 hw ether 08:00:27:51:05:a1
ifconfig enp0s3 up

# changing the MAC address
ip link set dev enp0s3 address 08:00:27:51:05:a3

## Network static configuration using Netplan (Ubuntu)

**# 1. Stop and disable the Network Manager**

sudo systemctl stop NetworkManager
sudo systemctl disable NetworkManager
sudo systemctl status NetworkManager
sudo systemctl is-enabled NetworkManager

**# 2. Create a YAML file in /etc/netplan**

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: false
      addresses:
```

```
        - 192.168.0.20/24
      gateway4: "192.168.0.1"
      nameservers:
        addresses:
          - "8.8.8.8"
          - "8.8.4.4"
```

**# 3. Apply the new config**
```
sudo netplan apply
```

**# 4. Check the configuration**
```
ifconfig
route -a
```

# OpenSSH

---

**# 1. Installing OpenSSH (client and server)**
```
# Ubuntu
sudo apt update && sudo apt install openssh-server openssh-client

# CentOS
sudo dnf install openssh-server openssh-clients

# connecting to the server
ssh -p 22 username@server_ip          # => Example: ssh -p 2267 john@192.168.0.100
ssh -p 22 -l username server_ip
ssh -v -p 22 username@server_ip       # => verbose
```

**# 2. Controlling the SSHd daemon**
```
# checking its status
sudo systemctl status ssh          # => Ubuntu
sudo systemctl status sshd         # => CentOS

# stopping the daemon
sudo systemctl stop ssh          # => Ubuntu
sudo systemctl stop sshd         # => CentOS

# restarting the daemon
sudo systemctl restart ssh          # => Ubuntu
sudo systemctl restart sshd         # => CentOS

# enabling at boot time
```

```
sudo systemctl enable ssh          # => Ubuntu
sudo systemctl enable sshd         # => CentOS

sudo systemctl is-enabled ssh          # => Ubuntu
sudo systemctl is-enabled sshd         # => CentOS
```

**# 3. Securing the SSHd daemon**
# change the configuration file (/etc/ssh/sshd_config) and then restart the server
```
man sshd_config
```

**a)** Change the port
```
Port 2278
```

**b)** Disable direct root login
```
PermitRootLogin no
```

**c)** Limit Users' SSH access
```
AllowUsers stud u1 u2 john
```

**d)** Filter SSH access at the firewall level (iptables)

**e)** Activate Public Key Authentication and Disable Password Authentication

**f)** Use only SSH Protocol version 2

**g)** Other configurations:
```
ClientAliveInterval 300
ClientAliveCountMax 0
MaxAuthTries 2
MaxStartUps 3
LoginGraceTime 20
```

# Copying files using SCP and RSYNC

---

## SCP

# copying a local file to a remote destination
```
scp a.txt john@80.0.0.1:~
scp -P 2288 a.txt john@80.0.0.1:~          # using a custom port
```

# copying a local file from a remote destination to the current directory
```
scp -P 2290 john@80.0.0.1:~/a.txt .
```

# copying a local directory to a remote destination (-r)

```
scp -P 2290 -r projects/ john@80.0.0.1:~
```

## RSYNC

```
# synchronizing a directory
sudo rsync -av /etc/ ~/etc-backup/

# mirroring (deleting from destination the files that were deleting from source)
sudo rsync -av --delete /etc/ ~/etc-backup/

# excluding files
rsync -av --exclude-from='~/exclude.txt' source_directory/ destination_directory/

# exclude.txt contents:
*.avi
music/
abc.mkv

rsync -av --exclude='*.mkv' --exclude='movie1.avi' source_directory/ destination_directory/

# synchronizing a directory over the network using SSH
sudo rsync -av -e ssh /etc/ student@192.168.0.108:~/etc-backup/

# using a custom port
sudo rsync -av -e 'ssh -p 2267' /etc/ student@192.168.0.108:~/etc-backup/
```

## WGET

```
# installing wget
apt install wget        # => Ubuntu
dnf install wget        # => CentOS

# download a file in the current directory
wget https://cdimage.kali.org/kali-2020.2/kali-linux-2020.2-installer-amd64.iso

# resuming the download
wget -c https://cdimage.kali.org/kali-2020.2/kali-linux-2020.2-installer-amd64.iso

# saving the file into a specific directory
mkdir kali
wget -P kali/ https://cdimage.kali.org/kali-2020.2/kali-linux-2020.2-installer-amd64.iso

# limiting the rate (bandwidth)
wget --limit-rate=100k -P kali/
https://cdimage.kali.org/kali-2020.2/kali-linux-2020.2-installer-amd64.iso
```

```
# downloading more files
wget -i urls.txt          # urls.txt contains urls

# starting the download in the background
wget -b -P kali/ https://cdimage.kali.org/kali-2020.2/kali-linux-2020.2-installer-amd64.iso
tail -f wget-log          # => checking its status

# getting an offline copy of a website
wget --mirror --convert-links --adjust-extension --page-requisites --no-parent http://example.org
wget -mkEpnp http://example.org
```

## NETSTAT and SS

---

```
# displaying all open ports and connections
sudo netstat -tupan
sudo ss -tupan
# checking if port 80 is open
netstat -tupan | grep :80
```

## LSOF

---

```
# listing all files that are open
lsof

# listing all files opened by the processes of a specific user
lsof -u username

# listing all files opened by a specific process
lsof -c sshd

# listing all files that have opened TCP ports
lsof -iTCP -sTCP:LISTEN
lsof -iTCP -sTCP:LISTEN -nP
```

## nmap

---

### Scanning hosts and networks using nmap

**##** SCAN ONLY YOUR OWN HOSTS AND SERVERS !!! **###**
**## Scanning Networks is your own responsibility ##**

**You can use scanme.nmap.org for safe scanning purposes.**

# Syn Scan - Half Open Scanning (root only)
nmap -sS 192.168.0.1

# Connect Scan
nmap -sT 192.168.0.1

# Scanning all ports (0-65535)
nmap -p- 192.168.0.1

# Specifying the ports to scan
nmap -p 20,22-100,443,1000-2000 192.168.0.1

# Scan Version
nmap -p 22,80 -sV 192.168.0.1

# Ping scanning (entire Network)
nmap -sP 192.168.0.0/24

# Treat all hosts as online,  skip host discovery
nmap -Pn 192.168.0.0/24

# Excluding an IP
nmap -sS 192.168.0.0/24 --exclude 192.168.0.10

# Saving the scanning report to a file
nmap -oN output.txt 192.168.0.1

# OS Detection
nmap -O 192.168.0.1

# Enable OS detection, version detection, script scanning, and traceroute
nmap -A 192.168.0.1

# reading the targets from a file (ip/name/network separated by a new line or a whitespace)
nmap -p 80 -iL hosts.txt

# exporting to out output file and disabling reverse DNS
nmap -n -iL hosts.txt -p 80 -oN output.txt

# Software Management (dpkg and apt)

---

## DPKG

```
# getting info about a deb file
dpkg --info google-chrome-stable_current_amd64.deb

# installing an application from a deb file
sudo dpkg -i google-chrome-stable_current_amd64.deb

# list all installed programs
dpkg --get-selections
dpkg-query -l

# filtering the output
dpkg-query -l | grep ssh


# listing all files of an installed package
dpkg-query -l | grep ssh
dpkg -L openssh-server

# finding to which package a file belongs
which ls
dpkg -S /bin/ls
    coreutils: /bin/cp

# removing a package
sudo dpkg -r google-chrome-stable

# purging a package
sudo dpkg -P google-chrome-stable
```

## APT

```
# updating the package index (doesn't install/uninstall/update any package)
sudo apt update

# installing or updating a package named apache2
sudo apt install apache2

# listing all upgradable packages
```

```
sudo apt list --upgradable

# upgrading all applications
sudo apt full-upgrade
sudo apt full-upgrade -y          # => assume yes to any prompt (useful in scripts)

# removing a package
sudo apt remove apache2

# removing a package and its configurations
sudo apt purge apache2

# removing dependencies that are not needed anymore
sudo apt autoremove

# removing the saved deb files from the cache directory (var/cache/apt/archives)
sudo apt clean

# listing all available packages
sudo apt list
sudo apt list | wc -l

# searching for a package
sudo apt list | grep nginx

# showing information about a package
sudo apt show nginx

# listing all installed packages
sudo apt list --installed
```

## Task scheduling using Cron

---

```
# editing the current user's crontab file
crontab -e

# listing the current user's crontab file
crontab -l

# removing the current user's crontab file
crontab -r
```

**## COMMON EXAMPLES ##**

```
# run every minute
* * * * * /path_to_task_to_run.sh

# run every hour at minute 15
15 * * * * /path_to_task_to_run.sh

# run every day at 6:30 PM
30 18 * * * /path_to_task_to_run.sh

# run every Monday at 10:03 PM
3 22 * * 1 /path_to_task_to_run.sh

# run on the 1st of every Month at 6:10 AM
10 6 1 * * /path_to_task_to_run.sh

# run every hour at minute 1, 20 and 35
1,20,35 * * * * /path_to_task_to_run.sh

# run every two hour at minute 10
10 */2 * * * /path_to_task_to_run.sh

# run once a year on the 1st of January at midnight
@yearly        /path_to_task_to_run.sh

# run once a month at midnight on the first day of the month
@monthly       /path_to_task_to_run.sh

# run once a week at midnight on Sunday
@daily         /path_to_task_to_run.sh

# once an hour at the beginning of the hour
@hourly        /path_to_task_to_run.sh

# run at boot time
@reboot        /path_to_task_to_run.sh
```

All scripts in following directories will run as root at that interval:

```
/etc/cron.hourly
/etc/cron.daily
/etc/cron.hourly
/etc/cron.monthly
/etc/cron.weekly
```

# Getting System Hardware Information

---

```
# displaying full hardware information
lshw
lshw -short       # => short format
lshw -json        # => json format
lshw -html        # => html format

# installing inxi
apt install inxi

inxi -Fx

# displaying info about the CPU
lscpu
lshw -C cpu
lscpu -J      => json format

# displaying info about the installed RAM memory
dmidecode -t memory
dmidecode -t memory | grep -i size
dmidecode -t memory | grep -i max

# displaying info about free/used memory
free -m

# getting info about pci buses and about the devices connected to them
lspci
lspci | grep -i wireless
lspci | grep -i vga

# getting info about USB controllers and about devices connected
lsusb
lsusb -v

# getting info about hard disks
lshw -short -C disk
fdisk -l
fdisk -l /dev/sda
lsblk

# installing hdparm
```

```
apt install hdparm
hdparm -i /dev/sda
hdparm -I /dev/sda

# benchmarking disk read performance
hdparm -tT --direct /dev/sda

# getting info about WiFi cards and networks
lshw -C network
iw list
iwconfig
iwlist wlo1 scan

# getting hardware information from the /proc virtual fs
cat /proc/cpuinfo
/proc/partitions
cat /proc/meminfo
cat /proc/version
uname -r      # => kernel version
uname -a

acpi -bi        # battery information
acpi -V
```

### Working directly with device files (dd)

```
# backing up the MBR (the first sector of /dev/sda)
dd if=/dev/sda of=~/mbr.dat bs=512 count=1

# restoring the MBR
dd if=~/mbr.dat of=/dev/sda bs=512 count=1

# cloning a partition (sda1 to sdb2)
dd if=/dev/sda1 of=/dev/sdb2 bs=4M status=progress
```

## Service Management using systemd and systemctl

---

```
# showing info about the boot process
systemd-analyze
systemd-analyze blame

# listing all active units systemd knows about
systemctl list-units
systemctl list-units | grep ssh
```

```
# checking the status of a service
sudo systemctl status nginx.service

# stopping a service
sudo systemctl stop nginx

# starting a service
sudo systemctl start nginx

# restarting a service
sudo systemctl restart nginx

# reloading the configuration of a service
sudo systemctl reload nginx
sudo systemctl reload-or-restart nginx


# enabling to start at boot time
sudo systemctl enable nginx

# disabling at boot time
sudo systemctl disable nginx

# checking if it starts automatically at boot time
sudo systemctl is-enabled nginx

# masking a service (stopping and disabling it)
sudo systemctl mask nginx

# unmasking a service
sudo systemctl unmask nginx
```

# Bash Programming

---

### Bash Aliases

```
# listing all Aliases
alias

# creating an alias:  alias_name="command"
alias copy="cp -i"
```

To make the aliases you define persistent, add them to ~/.bashrc

```
# removing an alias: unalias alias_name
unalias copy
```

**Useful Aliases**

```
alias c="clear"
alias cl="clear;ls;pwd"
alias root="sudo su"
alias ports="netstat -tupan"
alias ssh config="sudo vim /etc/ssh/sshd_config"
alias my_server="ssh -p 3245-l user100 80.0.0.1"
alias update=”sudo apt update && sudo apt dist-upgrade -y && sudo apt clean”
alias lt="ls -hSF --size -1"
alias ping='ping -c 5'

# interactive File Manipulation
alias cp="cp -i"
alias mv="mv -i"
alias rm="rm -i"
```

**Important alias**

```
# This may look a bit confusing, but essentially, it makes all of the other aliases you define function correctly when used with sudo.
alias sudo='sudo '      # use single quotes, not double quotes.
```

## Bash Variables

```
# defining a variable: variable_name=value
# variable names should start with a letter or underscore and can contain letters, digits and underscore
os="Kali Linux"
version=10

# referencing the value of a variable (getting the variable value): $variable_name
echo $os
echo $version

# defining a read-only variable (constant)
declare -r temperature=100

# removing (unsetting) a variable
unset version
```

```
# listing all environment variables
env
printenv

# searching for an environment variable
printenv PATH
env | grep -i path

# creating new environment variables for the user: in ~/.bashrc add export MYVAR=”value”
export IP="80.0.0.1"

# changing the PATH
export PATH=$PATH:~/scripts     # in ~/.bashrc

# getting user input
read MY_VAR
echo $MY_VAR

# displaying a message
read -p "Enter the IP address: " ip
ping -c 1 $ip

read -s -p "Enter password:" pswd
echo $pswd
```

## SPECIAL VARIABLES AND POSITIONAL ARGUMENTS

Run: ./script.sh filename1 dir1

```
$0 => the name of the script itself (script.sh)
$1 => the first positional argument (filename1)
$2 => the second positional argument (dir1)
...
${10} => the tenth argument of the script
${11} => the eleventh argument of the script

$# => the number of the positional arguments
"$*" => string representation of all positional argument
$? => the most recent foreground command exit status

#!/bin/bash
echo "\$0 is $0"
echo "\$1 is $1"
echo "\$2 is $2"
```

```
echo "\$3 is $3"
echo "\$* is $*"
echo "\$# is $#"

# Move to the script's directory and run it as: ./script_name.sh Ubuntu CentOS "Kali Linux"
"Windows 10"
```

## Program Flow Control (if..elif..else statements)

```
# if [ some_condition_is_true ]
# then
#   //execute this code
# elif [ some_other_condition_is_true ]
# then
#   //execute_this_code
# else
#   //execute_this_code
# fi
## Examples:

i=1
if [[ $i -lt 10 ]]
then
        echo "i is less than 10."
fi
##################
i=100
if [[ $i -lt 10 ]]
then
         echo "i is less than 10."
else
       echo "i is greater than or equal to 10."
fi
################
i=10
if [[ $i -lt 10 ]]
then
        echo "i is less than 10."
elif [[ $i -eq 10 ]]
then
        echo "i is 10"
else
        echo "i is greater than or equal to 10."
fi
```

## TEST CONDITIONS

man test

```
# For numbers (integers) ###
# -eq   equal to
# -ne   not equal to
# -lt   less than
# -le   less than or equal to
# -gt   greater than
# -ge   greater than or equal to

# For files:
# -s    file exists and is not empty
# -f    file exists and is not a directory
# -d    directory exists
# -x    file is executable by the user
# -w    file is writable by the user
# -r    file is readable by the user

# For Strings
# =     the equality operator for strings if using single square brackets [ ]
# ==    the equality operator for strings if using double square brackets [[ ]]
# !=    the inequality operator for strings
# -n $str   str is nonzero length
# -z $str   str is zero length

# &&  => the logical and operator
# ||  => the logical or operator
```

## For loops

```
#!/bin/bash
# iterating over a list of strings
for os in Ubuntu CentOs Slackware "Kali Linux"
do
        echo "os is $os"
done

# iterating over a list of numbers
for num in {3..7}
do
        echo "num is $num"
done
```

```
# iterating over a list of numbers in increments
for x in {10..100..5}
do
	echo $x
done


# iterating over a list of files
for item in ./* # files in the current dir
do
	if [[ -f $item ]]
	then
		echo "Displaying the contents of $item"
		sleep 1
		cat $item
		echo "########################"
	fi
done


# C/Java style
for ((i = 0 ; i <= 50 ; i++))
do
  echo "i = $i"
done
```

### While Loops

```
#!/bin/bash

i=0
while [[ $i -lt 10 ]]
do
	echo "i: $i"
   ((i++))     # same as: let i=i+1
done
```

### Case

```
#!/bin/bash
echo -n "Enter your favorite pet:"
read PET
```

```
case "$PET" in
        dog)
                echo "Your favorite pet is the dog."
                ;;
        cat | Cat)
                echo "You like cats."
                ;;
        fish | "African Turtle")
                echo "Fishes or turtles are great!"
                ;;
        *)
                echo "Your favorite pet is unknown!"
esac
```

## Functions

```
#!/bin/bash

# defining a function: method 1
function print_something () {  .
        echo "I'm a simple function!"
}

# defining a function: method 2
display_something () {
        echo "Hello functions!"
}
# neither of the above methods of specifying a function is valid.

# calling the functions
print_something
display_something
```

```
#!/bin/bash
create_files () {
          echo "Creating $1"
          touch $1
          chmod 400 $1

          echo "Creating $2"
          touch $2
          chmod 600 $2
```

```
}

# calling the function with 2 args
create_files aa.txt bb.txt

# function that returns a value (output of a command)
function lines_in_file() {
        grep -c "$1" "$2"
}

n=$(lines_in_file "usb" "/var/log/dmesg")
echo $n
```