Basic Linux Commands

cd ~ = sends you to home directory

cd 'name' = changes the directory to 'name'

mkdir 'name' = creates the 'name' directory

mkdir -p folder1/folder2/folder3 = creates nested folders inside each other

rmdir 'name' = remove the 'name' directory (if the directory is empty)

rm -r folder = removes folder and everything inside it (for nested folders)

rm 'name' = removes the file 'name'

rm * = deletes all file and directories in current directory

ls = lists all files in the current directory

ls -l = lists all files with permissions, ownership, group, size in current directory

ls -lh = same as above but shows size in kilobyte instead of bytes

ls -lh file1 = will show all properties about this file

ls -a = lists all files (including hidden files)

ls -la = mix of "ls -l" and "ls -a"

cd .. = go to parent directory

cd ../.. = go two directories back

cd - = show info of previous directory

\n = new line within paragraph

pwd = shows the current directory with it's full address

touch = create an empty file

d = directory / c = character device / b = buffred device / l = soft link to the file touch text = create a text file named 'text' (only if there is NOT a file named 'text' in this directory)

touch {babak, ali, hasan, omid} = will create four files with given names

touch file_{01..1000} = creates files file_01, file_02, ..., file_1000

touch text1 text2 text3 = create 3 text files named text1, text2, text3

echo Hello = print the 'Hello' on the screen

echo Hello>text = put the string 'Hello' inside a text file named 'text'

echo {1..10..2} = 1 3 5 7 9 | echo {1..10..3} = 1 4 7 10 | echo {A..Z} = A B C ... Z

cat text = show content of a text file named text on the screen

cat text text2 text3 = shows text files one after another

mv tex* dir1/subdir1/ = move any file in the current directory with starting name 'tex' to the other specified directory

? = every character is possible (but only one)

* = whatever character is possible

[] = a range of characters and numbers [A-Z] or [a-z] or [0-9]

mv filename1 filename2 = change name of file from first one to the second one

mv fileOne Directory1/ = moves a file to a directory

mv fileOne Directory1/fileTwo = moves file to a directory and gives it a new name

cp fileOne fileTwo = creates copy of a file

cp folder1/file1 folder2/fileCopy = copies a file and creates a copy of it in another folder

nano file.text = will open a text file in nano text editor (if doesn't exist it will create it)

ispell fileone = will open the text file and check all spells and show correct words if we have any mistakes sort textOne = sorts the textfile based on alphabet

< = is an input stream that gets data from a textfile or command : prog1 < file.txt

> = is an output stream that writes to a textfile : ls -a > textone : gets the list of files with permission in this folder and saves it inside a text file.

>> = appends text to another textfile (the first one must be already created) i.e : cat text|tr s S >> text

tr s S = this command will change all character s in the textfile to S, i.e : tr r R<text1>text2 : here we got text from text1 then changed all r to R then created text2 and added the text to it.

clear = clears the terminal from any written word

tree = will show the folder system tree of the current folder and its subfolder

diff fileOne FileTwo = compares two textfiles and when there is a difference it shows both lines in both files

passwd = with this command you an change the password for this user

how to create variable in terminal : variable=randomName

how to echo variable : echo $variable

info nameOfCommand = gives information about that command

if you press the 'up' arrow it will show all the prevoius written commands

if you want to write a file with space in it's name in the terminal use \ for spaces, i.e : best movies => best\ movies

exit = will exit the command prompt

| = gets output of one command and uses it as input of another command : cat diary | grep Bobby : it opens the textfile diary then searches for the word Bobby with grep command

prog1 | prog2 | prog3 = get output of program1 feed it to program2 then get output of program2 and feed it as input to program3
prog1 && prog2 = if the first command runs successfully then runs command two other wise it stops : ls -lh file1 && echo "file1 is here"

find log = finds all files that have the name 'log' in current directory

man ls = shows the manual page related to ls. you can leave that page by hitting key 'q'

more text1 = opens and shows the textfile inside the terminal

less text1 = opens and shows the textfile inside a reader application

date = shows the current date inside the terminal

date +"%d-%m-%y" = will show results like 17-02-2016

date +"%H-%M-%S" = shows hours-minutes-seconds

Ctrl+z = sends the current open program to background

bg = shows list of all programs in background

fg 'numberOfProgram' = sends the program with specific number to foreground (AKA maximize it)

how to open a shell script (.sh) file from terminal : bash name.sh

Ctrl+c = terminate the current program/command

tac text1 = displays text file in reverse order (end to start)

rw text1 = reads file from right to left (for arabic text)

cd = will take u back to your home directory

in linux use shift+insert for pasting (instead of Ctrl+p)

wget downloadLink = will download the file from internet to the current directory

GRUB is the GRand Unified Bootloader, a very powerful newish BootLoader that can be used to boot most operating system on the intel platforms

/sbin/ifconfig = shows the state of physical and virtual network configurations

soft links just link to another location on system. it doesn't have to always pin to another real file sudo -i = will change u from normal user to root (write 'exit' command to exit out of root access)

ls > /dev/null = will send results to empty place

INSTALL PROGRAMS

sudo = super user do

sudo apt-get update = updates the apt-get tool

how to check if u have already installed a program : nameOfProgram -version   : java -version or   php5 -version : if it is installed it will show the info about it.

sudo apt-get install nameOfPackage = this will install the package that you want. i.e : sudo apt-get install java

----

COMPRESS and DECOMPRESS files/folders

gzip file1 = compresses the file into file1.gz

gunzip file1.gz = decompresses the file1.gz back into file1

tar cvf name.tar folder1/ = this command will create a archive file named 'name.tar' from the folder 'folder1'

tar xvf name.tar = will extract the archived file

tar cvf Boby.tar file1 file2 file3 = this command will create a tar file named Boby.tar from 3 other files

----

HEAD and TAIL

head -n nr = displays the first number heading lines of text : ls -l | head -n 5

head -n -nr = displays all lines except the last nr lines : cat text.txt | head -n -5 (shows all lines except last 5 lines)

tail -n nr = displays the last nr lines : cat userLog | tail -n 5 (shows the last 5 lines) tail -n +nr = displays from the line having number nr to the end of file : ls -lah | tail -n +10 (shows lines from line 10 to the end)

WC -l filename = show number of lines in the file

find . -maxdepth|typed|wc -l = will show the number of file in the current directory

head file.txt = shows first few lines of the file

tail file.txt = shows last few lines of the file

GREP

grep = searches a string for some text or a word

grep "word" file = searches for a specific word / text inside a text file : grep "Bob" users.txt

grep "string" file_pattern :   grep "Bob" text* = will search for word Bob in all files that start with word text

grep -i "sTrIng" file = will do case-insensitive searching : grep "Bob" textfile : will search for all variations of Bob, boB, BoB, BOB,...

grep "REGEX" file : grep "usern*" users.txt = will search and find all strings like username, username1 , usernameX, ...

grep -w "string" file : grep -w "ok" stroy.txt = will only find full word ok and not okay or okaaay,....

grep -A nr "string" file : grep -A 5 "Bob" users.txt = will show nr lines after the first occurance of the word

grep -B nr "string" file : grep -B 3 "Joe" = will show nr lines before occurance of the word

grep -C nr "string" file : grep -C 2 "Claire" names.csv = will show 2 lines before and two line after the first occurance of word Claire

grep -c "string" file : grep -c "poor" economy.text = will show the number of times it found the word inside the text file.

grep -v "string" file : grep -v "good" story.txt = will show all the lines that do NOT contain the word 'good' wc -l = will show the number of lines : grep -v "good" story.txt | wc -l = shows number of the lines that do not have the word 'good'

grep -o -b "string" file : grep -o -b "randomWord" story.txt = will show position of the matched string inside the line that happened

grep -n "string" file : grep -n "teacher" list.txt : will show the number of each line where this word occures

----

CUT

cut -c1,5 textfile = shows from first to fifth character of each line of the text file

cut -b1,3 textfile = shows first and third character of each line of the file

cut -d " " -f1 = will only show the first field of text file (fields defined by space between them)

cut -d " " -f1,2 = same as above but selects both fields one and two

cut -d : -f1,3 = will select fields one and three (fields defined by : between them)

SED

sed -n 1,60p file.txt = will print lines 1 to 60 of text file on the screen

sed 5,10d file.txt = will hide lines 5 to 10 and show all other lines

sed -n /string/p file.txt = will only show lines that contain the string

sed -n /string1/,/string2/p file.txt = will show from the line that contains string1 and end with the line that contains string2

sed -n /^bob/p file.txt = will only show the lines that start with string bob

sed s/string1/string2/g file.txt = shows all the text but changes all string1 to string2 inside the file

sed 's/^/ /' file.txt = adds a space at the start of each line when showing the text

sed '2,5s/^/ /' file.txt = same as above but only for lines 2 to 5 sed '55,60s/^/\n/' file.txt = adds one line before lines 55 to 60

sed '10,14s/$/\n/' file.txt = adds one line after line 10 to 14

cat auth.txt | awk {'print $12'} = this will show the 12th phrase of each line on the screen (divided by space)

FIND

find <path> <search criteria> <action> = syntax of a search command

find . -type f -name abc.txt =
will only search for files, in the current directory (and the subdirectories inside it), with the name abc.txt

find / -type d = will only search for folders in all the hard drive

find . -type d -iname bob = will search in current directory for directories named bob in case-insensitive way

find . -type d -name "*.txt" = finds all files with extention .txt in current directory and it's sub-directories

find / -type f -perm 0777 = will find all files with 777 permission

find . -type f -perm 0755 -exec chmod 777 {} \;   = will find all files with permission 755 in current folder and change their permission to 777,   -exec = means execute

find . -mtime +1   = finds all files and folders that were modified in more than one day before (more than 24 hours)   find . -mtime -1 = modified less than one day ago (in last 24 hours)
find . -mmin -30 = modified less than 30 minutes ago

find . -type f -mmin +10 -mmin -30 = finds all files in current directory that were modified in more than 10 mins and less than 30 minutes ago

find / -size -1M = finds all files and folders that their size is smaller than one megabyte

find / -size +10M -size -30M = finds all files,folders with size bigger than 10mb and smaller than 30mb find . -maxdepth 1 -name "*.txt" = will only find files in current directory (and NOT sub-directories)

find . -maxdepth 1 -name "*.txt" = will only find files in current directory (and NOT sub-directories)

LINK

ln -s addressOfFile addressOfLink = creates a soft link between two files/directories ln addressOfFile addressOfLink = creates a hard link

ln -s abc.txt abc_soft.txt

links are deleted like normal files with "rm" command

soft link = is just a pointing system like hyperlinks in web, it will not work if you delete the actuall file

hard link = is actually a exact copy of the original file, even if you delete the file it will still be available

----

USERS

whenever you create a new user, linux automatically creates a group for that user with the same name.

cat ./etc/passwd = shows list of all users, only who have /bin/bash in the end are real users

cat ./etc/group = shows list of all groups

cat ./etc/shadow = shows password for all users (encrypted)

ls /home/ = lists home directory for all users of this system

sudo useradd name = adds a new user to the system

sudo useradd -m -d /home/customName name = adds user to the system with custom home directory name

sudo passwd name = changes/gives a new password to the user sudo userdel username = will delete the user from this system

sudo usermod -L username = locks the account so the user can not login to his account

sudo usermod -U username = unlocks the user account

GROUP

sudo groupadd groupName = creates a new user group

sudo usermod -a -G groupName username = adds the user to the group, usermod = user modification, -a = append, -G = group

if we use -G = it will add this user to a new group, if we use -g = it will remove user from all other groups and add him here.

if you go two directories back from home (cd ../..) and from there to (cd ./etc/passwd) you can see all the usernames inside the passwd file (cat passwd)

SSH

how to connect to server computer through ssh : ssh username@ipaddress   i.e :
ssh root@52.49.96.196

then write yes (since it's first time connecting to this system) and then enter password to connect to the server computer.

passwords for websites can be hacked but SSH keys cannot be hacked.

ssh-keygen -t rsa = this will generate two ssh codes (one for your computer and one for the public key)

ssh-copy-id username@ipaddress = this command will set the ssh key for your server, after this when u want to log in the server it will automatically check ssh key on server with your private ssh and you don't need to write password.

nano /etc/ssh/sshd_config = this will open the configuration files about the ssh with the nano text editor, inside the file, there is a section 'PermitRootLogin' change 'Yes' to 'without-password', then press Ctrl+x to save and quit the config file. from now the users can only log in to the server with ssh.

reload ssh = write this command whenever you make changes to ssh so it will be updated.

SFTP

sftp is a combination of ftp and ssh, you can transfer files from your own computer to the server through the ssh protocol.

Upload file/folders to server :

1- sftp username@ipaddress : sftp root@52.19.92.15 = this will start the sftp connection to transfer files

2- cd ../.. = go to directories back so that you are in the root directory of the server, if you do ls -la you should see folders with root owner

3- cd var/www/html = go to html folder, this is where all the files that will be shown in your website are saved at.

4- put addressOfFileOnYourPc : put Desktop/index.html = this command will put files from your pc to the current folder inside the server (when you are in sftp mode)

for puting a directory inside the server :
1- createa directory(with same name) in server mkdir directroyName

2- put -r addressOfFolder : put Desktop/books = this will put everything inside that folder in the new folder that we created on server

Download file/folders from server :

get nameOfFile addressTobeSaved/newName : get index.html Desktop/random.html = will save the file from server to custom directory in our computer

get -r nameOfFolder addressTobeSaved/newName : get -r html Desktop/randomFolder = will save entire content of a directory on server to new directory on our computer
----

PARTITIONS and HARD DRIVE

hard drives and devices on linux are called like : sda, sdb, sdc,....

each partion of a hard drive is called like : sda1, sda2 or sdb1, sdb2,...

file formats exclusive to linux are ext3, ext4

fat32, nfts are cross platform file formats for both linux and windows

how to create partions :

1- sudo apt-get install gparted = it is a visual program to create partions

2- sudo gparted = open the program

3- unmount the current partion and delete it

4- add new partion, align to mib and as primary partion. for partion that you want to mount an linux operating system use ext4 and for the one to use on both pc/linux use nfts or fat32

PROCCESS

a process is any running program or command

ps = will show all the procceses running right now

PID = is the number associated with this process

TIME = is the time it took for cpu to run this process

when you run a command in terminal you can't run any other command untill the first process is done

Ctrl+c = terminates the running process

command & = will run the process in the background so you can run other commands in terminal : xlogo &

a job is another kind of PID

jobs = will show all running jobs with their job number

fg %job number = sends a job from background to foreground, i.e : fg %1

Ctrl+z = pause / stops a running process, you can not kill a process while it's paused

kill PID = will terminate a process

kill 9 PID = terminates a process immediately (use with caution), 9 here is a immediate kill signal

sudo killall proccessName = will kill all the processes with this process name

pkill proccessName = sends kill signal to all the processes that have full or part name of this proccessName

kill -STOP PID = will pause a process

kill -CONT PID = will continue the process

top = lists all processes with their usage of cpu and memory

init is the first process in linux system, always have 1 pid

all processes except init have parent process. parent process clones (forkes) itself and the clone process calls the needed child process to run

ps aux | grep proccessName = will search and list all proccesses related to that process name

4 states of a proccess :

1- runnable : it can run the next time cpu has some free percentage
2- sleeping : the process is waiting for something (like user input or cd-rom input)
3- zombie : the process is finishing what it's doing and is waiting to give back the results and be kiled
4- stopped : a process that has been paused by a signal by another process or the user and is wating to be continued

sudo apt-get install htop = installs htop which is a better version of top application to monitor proccesses

Niceness of proccesses :

when proccesses have high proirety they will be the most important procces for cpu/memory and vice versa

highest proierty : -20   lowest proierty : 19

if you set a proccess's niceness to 20 it will freeze the linux system since it will consume all power of the cpu

sudo renice -5 PID = will reset the niceness of a running proccess to -5 (high proirety)

nice -n 15 nameOfProccess = will lunch the process with nicness of 15 (low proirety)

/proc = is the place that system saves information about all running proccesses in different folders

ls -l /proc = will show list of all folders inside proc, each of them is named after a PID (number of a proccess)

ls -l /proc/1/ = shows info inside the pid one folder (init proccess is always 1), comm = name of proccess , cmdline = address of where the proccess is being executed
----
how to check the HASHED file :

when u download a file from internet if the file that u get is the same as the one on the site they both sould have the same hash code. (otherwise some hacker is sending u a virus file)

for SHA-1 hash write 'sha1sum nameoffile' command and it will give the hash code for the file.

comparing the hash of a file with given hash :
sha1sum bloodyFile | grep randomhash1213dfkjshjf43242lknf
if it returned the searched (greped) hash text then it is the same file.

CHMOD

chmod = change mode

chmod code nameOfFile

3 digit code = user+group+others

read = 4   write = 2 execute = 1

7 = read+write+execute   0 = no permission

chmod 777 nameOfFile = means that everyone can do what ever they want

chmod 444 nameOfFile = means that all users only have permission to read

when u type ls -a the shown files either start with - or d : - = means file   d = means directory
r = read w = write x = execute

rwx = permission to read, write, execute

r-- = permission to read only   rw- = permission to read and write only   --x = permission to execute only

example : -rwxrw-r-- : this is a file, read/write/exceute for owner, read/write for group, read-only for all others (i.e : permission to change a webpage in a website)

u = user
g = group
o = other people

how to add permission : chmod o+w fileOne : this will add the writing permission to all other people for fileOne

how to remove permission : chmod o-w fileOne : this will remove the writing permission from all other people for fileOne

chmod g-w-x fileOne : remove permission to write and execute from group

----
CRON

crontab -l = lists all crons that this user has created

crontab -e = edit / create cron

(minute) (hour) (day of month) (month of year) (week day) (the command to be executed) = system of writing a cron job minute = 1 to 60   hour = 1 to 24   day = 1 to 30   month = 1 to 12   week day = 1 to 7

* * * * * echo "hi there" >> /home/bob/Desktop/text.txt = will add a string to the text file every minute of hour of day of month and weekday (basically every minute)

sudo ls var/spool/crontab/ = will show list of all cron jobs for this user

15 10 1-10/2 * 5 echo "everything looks good?" >> /home/bob/Desktop/filetext.txt = min 15 of hour 10 of days 1,3,5,7,9 of every month and every friday(5 of 7) append the string to this address

* 12 12,13,14,15 * 3,7 = every minute of hour 12 of days 12,13,14,15 of every month and days 3 and 7 of the week

/etc/cron.allow = here you can ban users from creating crons

if you root you can edit an spesific user's cron : crontab -e -u username
----
PRINTF

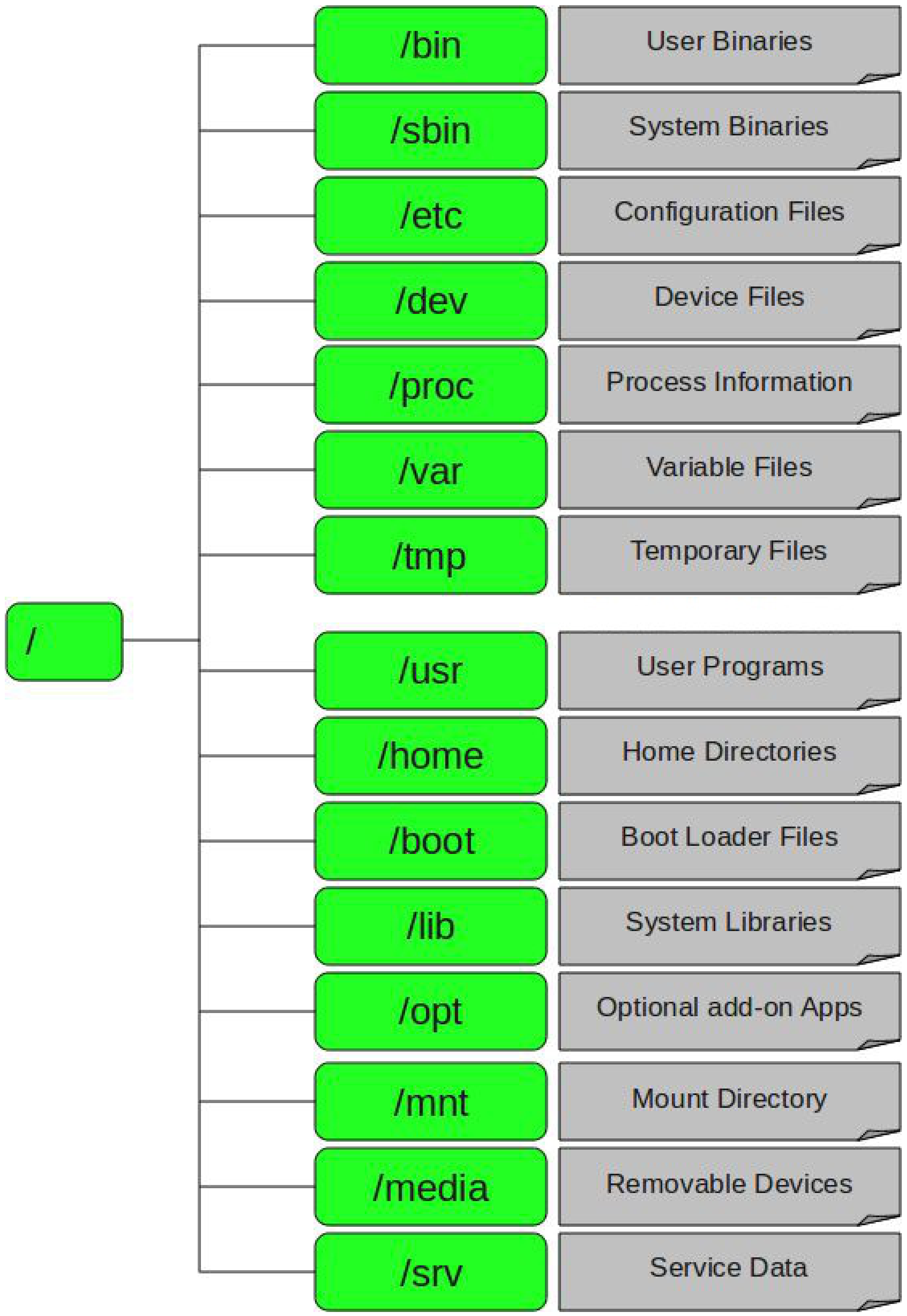
printf "Name:\t%s\nID:\t%04d\n" "Bob" "12" : \t = tab   %s = first variable   \n = next line   %04d = second variable, must be 4 characters

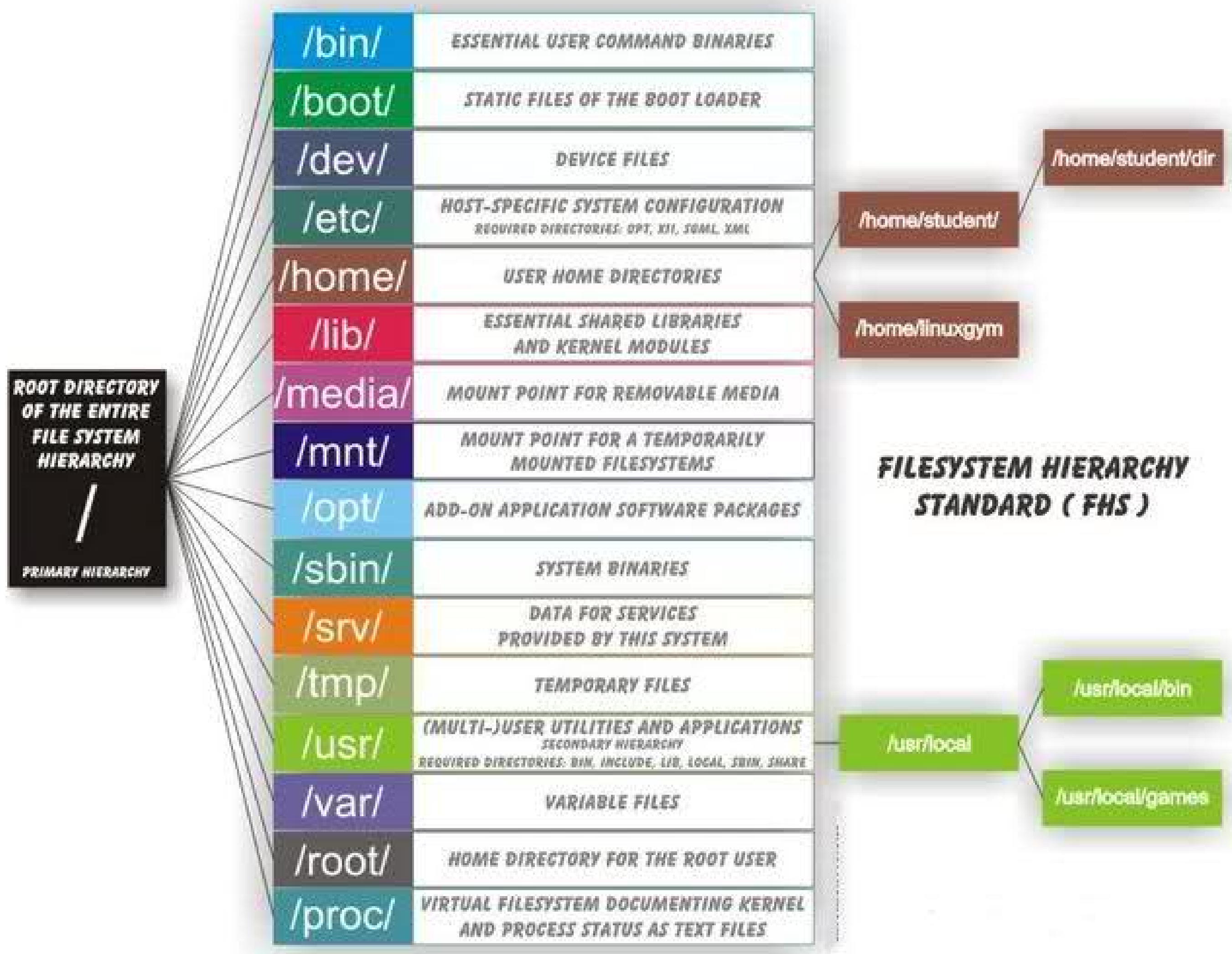
```
Name:	Bob
ID:	0012
```

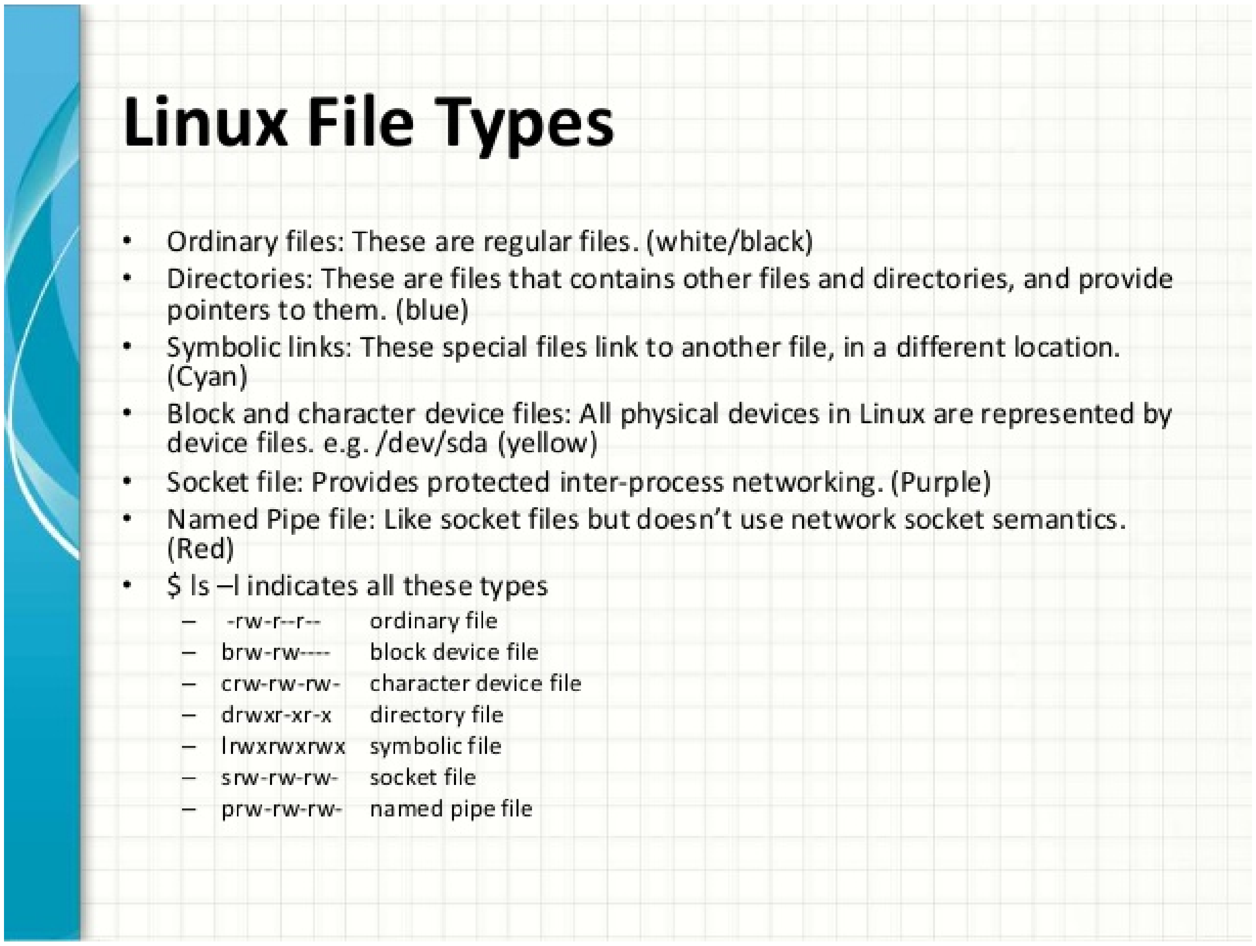
----
sudo ifconfig = gives info about all network adapters and their ip and mac addresses

sudo dhclient = it will release and renew the ip address

sudo /etc/init.d/networking restart = this command will restart the network setting (use it if you have changed the network settings)


/
/bin
User Binaries
/sbin
System Binaries
/etc
Configuration Files
/dev
Device Files
/proc
Process Information
/var
Variable Files
/tmp
Temporary Files
/usr
User Programs
/home
Home Directories
/boot
Boot Loader Files
/lib
System Libraries
/opt
Optional add-on Apps
/mnt
Mount Directory
/media
Removable Devices
/srv
Service Data


ROOT DIRECTORY OF THE ENTIRE FILE SYSTEM HIERARCHY
/
PRIMARY HIERARCHY
/bin/
ESSENTIAL USER COMMAND BINARIES
/boot/
STATIC FILES OF THE BOOT LOADER
/dev/
DEVICE FILES
/etc/
HOST-SPECIFIC SYSTEM CONFIGURATION
REQUIRED DIRECTORIES: OPT, X11, SGML, XML
/home/
USER HOME DIRECTORIES
/lib/
ESSENTIAL SHARED LIBRARIES AND KERNEL MODULES
/media/
MOUNT POINT FOR REMOVABLE MEDIA
/mnt/
MOUNT POINT FOR A TEMPORARILY MOUNTED FILESYSTEMS
/opt/
ADD-ON APPLICATION SOFTWARE PACKAGES
/sbin/
SYSTEM BINARIES
/srv/
DATA FOR SERVICES PROVIDED BY THIS SYSTEM
/tmp/
TEMPORARY FILES
/usr/
(MULTI-)USER UTILITIES AND APPLICATIONS
SECONDARY HIERARCHY
REQUIRED DIRECTORIES: BIN, INCLUDE, LIB, LOCAL, SBIN, SHARE
/var/
VARIABLE FILES
/root/
HOME DIRECTORY FOR THE ROOT USER
/proc/
VIRTUAL FILESYSTEM DOCUMENTING KERNEL AND PROCESS STATUS AS TEXT FILES
/home/student/
/home/student/dir
/home/linuxgym
/usr/local
/usr/local/bin
/usr/local/games
FILESYSTEM HIERARCHY STANDARD (FHS)



Linux File Types

- Ordinary files: These are regular files. (white/black)
- Directories: These are files that contains other files and directories, and provide pointers to them. (blue)
- Symbolic links: These special files link to another file, in a different location. (Cyan)
- Block and character device files: All physical devices in Linux are represented by device files. e.g. /dev/sda (yellow)
- Socket file: Provides protected inter-process networking. (Purple)
- Named Pipe file: Like socket files but doesn't use network socket semantics. (Red)
- $ ls –l indicates all these types

-rw-r--r--	ordinary file
brw-rw----	block device file
crw-rw-rw-	character device file
drwxr-xr-x	directory file
lrwxrwxrwx	symbolic file
srw-rw-rw-	socket file
prw-rw-rw-	named pipe file

Bash Scripting

SHELL SCRIPTS

how to create a shell script :

1- open a new documnet file in any text editor

2- #!/bin/sh = this should be the first line a shell script, it shows the linux it is shell file

3- # = any thing after this is a comment and will be ignored

how to run a shell script :

bash nameOfScript : bash Bobcript   = this will run the script (if it has excute permission)

1- set the script to executable : chmod +x script.sh   2- run the script : ./script.sh

echo -n "Hello"; echo " World" = -n wil make sure we don't go to the next line

echo "$x is my thing " = apple is my thing   echo '$x is my thing' = $x is my thing
----

echo

```
#!/bin/bash
# basic shell script

#ls

a=babak
b="Habib nejad"
c=125

echo $a
echo $b
echo $c

declare -i d=123     #d is an integer
declare -r e=1000    #e is an read-only and can not be modified
```

```
echo $MACHTYPE     # shows type of the system
echo $HOSTNAME     # shows the host name
echo $BASH_VERSION # shows current version of bash software
echo $0            # shows name of the running script



# Command substitution :

d=$(pwd)
echo $d

s=$(ls -l)
echo $s

# this command is for getting ping from a website :
a=$(ping -c 1 example.com | grep 'bytes from' | cut -d = -f 4)
echo $a
```
----

mathematical operations

```
#!/bin/bash

$((a+b))   # typical form of a mathematical operation in bash


d=2
e=$((d+2))
echo $e

((e++))
echo $e

((e--))
echo $e

((e+=3))
echo $e

((e-=6))
echo $e
```

```
((e*=4))
echo $e

((e/=2))
echo $e

f=$((1/3))
echo $f    # bash only works with integers
```
----

comparing

```
#!/bin/bash

# [[  ]]    1:false   0:true

[[ "cat" == "cat" ]]
echo $?

[[ "cat" = "dog" ]]     # both = and == work for compring strings
echo $?

[[ 20 -lt 100 ]]      # -lt = less than
echo $?

[[ 25 -ge 25 ]]       # -gt = greater or equal >=
echo $?


a=""
b="catDog"

[[ -z $a && -n $b ]]     # -z : check if the string is null        -n : check if string is not null
echo $?
```
----

editing strings

```
#!/bin/bash

a="hello"
b=" World"
```

c=$a$b   # add two strings together
echo $c

# lenght of the string ${#a}
echo ${#a}
echo ${#c}

d=${c:4}   # the $c variable from its forth character
echo $d

e=${c:3:4}   # start at third character and show 4 characters after that
echo $e

f=${c: -4} # shows last 4 characters of string (backward)
echo $f

fruit="cherry banana apple tangerine banana cucumber banana orange"
echo ${fruit/banana/mouz} # this will change first instnce of banana with mouz in variable fruit

echo ${fruit//banana/mouz} # change all instances of banana with mouz in variable fruit

echo ${fruit/#cherry/gilas} #only replaces if the searched term is first inisde the variable

echo ${fruit/o*/sib}   #all words that start with o
----
arrays

#!/bin/bash

a=()
b=("apple" "cherry" "orange")

echo ${b[1]}

b[5]="kiwi"   # you don't need to populate every element of an array

b+=("mango") # adds to the end of an array

echo ${b[6]}

echo ${b[@]}   # show all elements of an array

```
echo ${b[@]: -1}   # shows last element of an array

# array with key-value pairs :

declare -A myArray

myArray[color]=blue
myArray["my office"]="los angles"   # if the key or value have space in them you should you ""

echo ${myArray[color]}
echo "the city that I live in is ${myArray["my office"]}"
```

```
#!/bin/bash

cat  << EndOfText    # this is a  sign to show end of text and will not be shown  when you run the
script
fdfsdfdsf
dsfsdfsdf
dsfsdfsdf
EndOfText



cat <<- EnditBro    # the - after << removes the tabs from start of lines when showing them on
the screen
        dsfsdfsf
                weruwepruwe
toprtuoieurt
                zxcnv,zmv
EnditBro
```
----

if

```
#!/bin/bash

a=3
if [ $a -gt 4 ]; then
        echo "$a is greater than 4"
else
        echo "$a is not greater than 4"
fi
```

```
b="44this is my string!"
# [0-9]+ : is a regular expression and means if there is one or more number inside the string
if [[ $b =~ [0-9]+ ]]; then
	echo "There is a number inside the string : $b"
else
	echo "there is no number inside the string : $b"
fi
```
----

while loop

```
#!/bin/bash

# while loop works when the expression is true
i=0
while [ $i -le 10 ]; do
	echo i: $i
	((i++))
done


# until loop  works when the expression is false
j=0
until [ $j -ge 10 ]; do
	echo j: $j
	((j++))
done
```
----

for loop

```
#!/bin/bash

for i in 1 2 3 4 5
do
	echo $i
done


for j in {1..10..2}
do
```

```
        echo $j
done


for (( z=0; z<=10; z++ ))
do
        echo $z
done


arr=("apple" "banana" "cherry")
for x in ${arr[@]}
do
        echo $x
done


declare -A arrr
arrr["name"]="Bob Habib"
arrr["id"]="1031B"
# ! in ${!arrr[@]} means that $v is the key and not the value
# we put array element inside "" since there may be space in the string
for v in "${!arrr[@]}"
do
        echo $v: ${arrr[$v]}
done
------------------------------------------
```

case

```
#!/bin/bash

a="cat"

case $a in
        cat) echo "it's a cat yo!";;
        dog|puppy) echo "it's a dog yo";;   # it will accept either dog or puppy
        *) echo "no match brah";;
esac
```

----
function

```
#!/bin/bash

function greeting {
	echo "Hi there $1"
	echo "it's a nice $2 aint it?"
}

# how to call a function :
greeting Bob Day
greeting Homie shit


# function that accepts array/list
function numberThings {
	i=1
	for f in $@; do     # $@ means all elemnts inside the list / array
		echo $i: $f
		((i+=1))
	done
}

numberThings $(ls)     # passing command to the function

numberThings apple pineapple applePie
```

----
send data to script

```
#!/bin/bash

echo $1
echo $2


# when u want to send a list/array of argument to the script:
for i in $@
do
	echo $i
done
```

```
echo "number of arguments: $#"     #  $# shows number of entered arguments
```
----

send data with flag

```
#!/bin/bash
# flags

# u: and p: means you write -u/-p and some data after them.
# in the case if you entered -u before data it will asign it to username and if it was -p it will be password
# $OPTARG  is the data that you enter

while getopts u:p: option; do
        case $option in
                u) user=$OPTARG;;
                p) pass=$OPTARG;;
        esac
done

echo " $user  /  $pass "
```
----

send data with read

```
#!/bin/bash

echo "what is your name?"

# read command waits for user's input
read name

echo "what is your password?"

# read -s : it will not show what user has typed
read -s pass

# using read without echo command with read -p , it will do everything in one line
read -p "what is your favorite animal?  " animal

echo "name : $name, pass : $pass, animal: $animal"
```

```
#this is a select menu and user enters the number of their selected item :
select animal in "cat" "dog" "bird" "fish" "reptile"
do
        echo "you selected $animal"
        break
done
------------------------------------------------
```

check read data

```
#!/bin/bash

read -p "favorite animal? " a

#this will repeat the question untill user actually types something
while [[ -z $a ]]; do
        read -p "favorite animal? " a
done
echo "$a was selected"


read -p "what year [nnnn] ? " b

# here we check if user wrote a 4-digit number with regular expressions:
while [[ ! $b =~ [0-9]{4} ]]; do
        read -p "enter a valid year [nnnn] ? " b
done
echo "you selected year $b"

=========================================
```