**All commands and Tips**

**from TP1 to TP7**

| * To make the file **executable** :  1. **touch [fileName]** 2. **chmod +x [fileName]** 3. **./[fileName]**  * **while** statement **:**   **while command**  **do**  **Statement to be executed**  **done**  **-> we can have more that a command to execute before entering the loop**   * **For** statement :   **for var in word1 word2 ...wordn**  **do**  **Statement to be executed**  **done** | * **if** statement :   **if [ expression ]**  **then**  **statement1**  **else**  **statement2**  **fi**   * **Case** statement :   **case [Variable] in**  **[Pattern] ) Statement 1;;**  **[Pattern] ) Statement n;;**  **esac** |
| --- | --- |

**--------------------------------------------------------------------**

**TP1 : Manage/Query kernel and kernel modules at**

**runtime**

**--------------------------------------------------------------------**

**—-------- Commands used —--------**

* [**lsmod**](https://linuxize.com/post/lsmod-command-in-linux/): lists the loaded kernel modules
* [**uname**](https://linuxize.com/post/uname-command-in-linux/#:~:text=The%20uname%20command%20is%20used%20to%20print%20basic%20system%20information,free%20to%20leave%20a%20comment.): displays the information about the system
* [**rmmod**](https://linuxize.com/post/rmmod-command-in-linux/) **<moduleName>** or [**modprobe**](https://linuxize.com/post/modprobe-command-in-linux/) **-r <moduleName>** : unload the module
* [**insmod**](https://www.computerhope.com/unix/insmod.htm) **<FullModulePath>** or [**modprobe**](https://linuxize.com/post/modprobe-command-in-linux/) **<ModuleName>** : loads the kernel module but the main difference between the two commands is that modprobe command loads the module with its dependencies (if it needs another module it downloads it as well )

**—-------- Other Tips and commands —--------**

* To read columns separately from a file :

**while** **read [Param1] [Param2]** ...

**do**

**[Traitement]**

**done** < **[File\_Name]**

* To execute a command and store the output in a variable :

**var**=**$(command)**

* [**sed**](https://linuxize.com/post/how-to-use-sed-to-find-and-replace-string-in-files/) **-i 's/SEARCH\_REGEX/REPLACEMENT/g' INPUTFILE :**

**-i -** By default, sed writes its output to the standard output.

**s -** The substitute command, probably the most used command in sed.

**/ / / -** Delimiter character. It can be any character

**SEARCH\_REGEX -** Normal string or a regular expression to search for.

**REPLACEMENT -** The replacement string.

**g -** Global replacement flag. By default, sed reads the file line by line and changes only the first occurrence of the SEARCH\_REGEX on a line. When the replacement flag is provided, all occurrences are replaced.

**( instead of g we can have d which is used to Delete the pattern space and immediately start the next cycle. With this option we don’t specify the replacement** '/**[SEARCH\_REGEX]**/**d**' **)**

**INPUTFILE -** The name of the file on which you want to run the command.

* **head -$numS | tail -1 :** get the line num **$numS**

**--------------------------------------------------------------------**

**TP2 : Manage users and group accounts and related**

**system files**

**--------------------------------------------------------------------**

**—-------- Commands used —--------**

* **sudo** [**useradd**](https://linuxize.com/post/how-to-create-users-in-linux-using-the-useradd-command/) **[options] [username] :** adds a new user
* **sudo** [**userdel**](https://linuxize.com/post/how-to-delete-users-in-linux-using-the-userdel-command/) **[options] [username]** : deletes an existing user

**-r (--remove) :** deletes the user’s home directory

**note : you have to execute : sudo killall -u username so**

**the command works**

**-f (--force) :** forces the delete

* **sudo** [**usermod**](https://linuxize.com/post/usermod-command-in-linux/) **[options] [USER] :** modifies an existing user

**-a -G GROUP** : adds groups to the user

**-g GROUP :** change the primary group

**-c "GECOS Comment" :** change the user’s info

**-d HOME\_DIR :** change the home directory

**-s SHELL :** change the user’s default shell

**-u UID :** change the user’s id

**-l NEW\_USER :** change the user’s name

* [**groups**](https://linuxize.com/post/how-to-list-groups-in-linux/) **[username] :** lists the groups to which the user belongs **( default : list the current user’s groups )**
* [**gpasswd**](https://man7.org/linux/man-pages/man1/gpasswd.1.html#:~:text=The%20gpasswd%20command%20is%20used,of%20group%20administrators%20and%20members.) **[option] group :** administer /etc/group and /etc/gshadow

**-a, --add user :** Add the user to the named group.

**-d, --delete user :** Remove the user from the named group.

**—-------- Other Tips and commands —--------**

* [**getent**](https://man7.org/linux/man-pages/man1/getent.1.html) **[option][... database] [key ...] :**

**The most interesting databases :**

**group :** enumerate the group database

**passwd :** enumerate the passwd database

**[key ...] :** to search

* **getent passwd [username] :** displays the ligne in which we find the username in the **/etc/passwd** file
* [**column**](https://man7.org/linux/man-pages/man1/column.1.html) **[options] [file ...] :** columnate lists

**-t, --table :** Determine the number of columns the input contains and create a table. Columns are delimited with whitespace, by default,or with the characters supplied using the **--output-separator** option. Table output is useful for pretty-printing.

* **awk [options] 'program' [input-file...] / awk [options] [program-file...] [input-file...]**

**-F fs** To specify a file separator.

**-f file** To specify a file that contains awk script.

**-v var=value** To declare a variable.

**example : awk -F: '{print $1}' /etc/passwd**

* **cut OPTION... [FILE]... :**

**-f (--fields=LIST) -** Select by specifying a field, a set of fields, or a range of fields. This is the most commonly used option.

**-d (--delimiter) -** Specify a delimiter that will be used instead of the default “TAB” delimiter.

**example : cut test.txt -d ':' -f 1,3**

**--------------------------------------------------------------------**

**TP3 : Boot the system**

**--------------------------------------------------------------------**

**—-------- Commands used —--------**

* **who -r :** display the current runlevel
* **ps -ef | more :** display all the boot processes launched
* **ls /etc/rc$runlevel.d** : list the repositories containing the applications / programmes executed according to the runlevel **$runlevel**

**—-------- Other Tips and commands —--------**

* **ls \*.conf :** lists the config files
* **/etc/init.d :** is a directory and is not found in ubuntu
* **/etc/inittab :** is a file which doesn’t exist in ubuntu anymore
* **commande gestion des services :**

**systemctl** **ACTION <Nom\_du\_service>.service**

Où

* **ACTION** sera la commande que l'on souhaite appliquer à la dite unité:
  + **start :** démarrer le service
  + **stop :** arrêter le service
  + **restart :** relancer le service
  + **reload :** recharger le service
  + **status :** connaître l'état du service
* **<Nom\_du\_service>** est le nom du service visé.

Quelle que soit l'action menée sur un service, au prochain démarrage de la machine celui-ci devrait retrouver le status qui lui a été [défini par défaut](https://doc.ubuntu-fr.org/systemd#modifier_l_execution_d_un_service).

**--------------------------------------------------------------------**

**TP4 : Managing Software and Processes**

**--------------------------------------------------------------------**

**—-------- Commands used —--------**

* **ps -ef** : get information about the processes running within your system.

**-e :** Select all processes. Identical to **-A.**

**-f :** Do full-format listing

**--pid pidlist :** Select by process ID. Identical to **-p** and **p**

**--ppid pidlist :** Select by parent process ID.

**--Group grplist :** Select by real group ID (RGID) or name. Identical to **-G.**

**-C cmdlist :** Select by command name.

**-U userlist :** Select by real user ID (RUID) or name. It selects the processes whose real user name or ID is in the userlist list.

* **kill -l :** lists the kill command options
* **kill [options] <pid> [...] :** send a signal to a process

**<pid> [...]:** Send signal to every <pid> listed.

**-<signal> \ -s <signal> \ --signal <signal> :** Specify the signal to be sent (can be specified by using name or number)

**-l, --list [signal]:** List signal names.

* **1 (HUP)** - Reload a process.
* **9 (KILL)** - Kill a process.
* **15 (TERM)** - Gracefully stop a process.
* **19 (STOP)** - stops a process
* **18 (CONT) -** continues a process

**EXAMPLE**

* **kill -9 -1 :** Kill all processes you can kill.

**—-------- Other Tips and commands —--------**

* [**grep**](https://linuxize.com/post/how-to-use-grep-command-to-search-files-in-linux/) **[OPTIONS] PATTERN [FILE...] :** allows you to search for a character string in a file

**OPTIONS -** Zero or more options. Grep includes a [number of options](https://www.gnu.org/software/grep/manual/grep.html#Command_002dline-Options) that control its behavior.

**PATTERN -** Search pattern.

**FILE -** Zero or more input file names.

**-v** print lines not containing the string

**-c** count the number of lines containing the string

**-n** each line containing the string is numbered

**-xl** igne exactly matching the string

**-l** display the name of the files that contain the string

* **Regular expressions :**

| * **^** start of line * **.** any character * **$** end of line * **x \*** zero or more occurrences of the character x * **\ {n \}** to define the number of repetitions n of the character placed in front | * **x +** one or more occurrences of the character x * **x?** a single occurrence of the character x * **[...]** range of characters allowed * **[^ ...]** range of prohibited characters |
| --- | --- |

**Examples :**

* **[az] [az] \*** : searches for lines containing at least one lowercase character. **[az]** character allowed, **[az] \* match** search of letters allowed.
* **^ [0-9] \ {4 \} $ :** has the meaning, from start to end of file **$** , matches 4-digit numbers **[0-9] \ {4 \}** .
* **PATH=$PATH:.** : On a ajouté '.' à la variable PATH pour pouvoir exécuter le script sans './'
* **/dev/pts/[ID] :** contains the terminals and **[ID]** correspond to the **terminal ID**. To get it , execute the **ps** command and you’re going to find it in the **TTY** column
* **sudo p2 > /dev/pts/1 :** redirect the p2 script’s execution in the terminal with process number 1
* [**pidof**](https://man7.org/linux/man-pages/man1/pidof.1.html) **<process\_name> --** find the process ID of a running program.
* Ouvrir deux terminaux ---> **ps -ef > f1** (dans le terminal 1)---> fermer le terminal 2 ---> **ps -ef > f2** (dans le terminal 1) ---> **diff f1 f2** (dans le terminal 1) : ça nous permet de déterminer le PID du process du terminal 2

**--------------------------------------------------------------------**

**TP5 : Maintain an effective data backup strategy**

**--------------------------------------------------------------------**

**—-------- Commands used —--------**

* **Backup** :
* **tar** : **tar -cvf <archive> <fichier(s)>**

**-c :** creates an archive

**-v :** verbose mode

**-f :** tar will use the file archive as the tar archive it performs operations on, rather than tar’s compilation dependent default.

* **cpio** : **find <fichier> -depth | cpio -ov > <archive>**

**-o, --create :** Copy-out. Read a list of file names from the standard input and create on the standard output (unless overridden by the **--file option**) an archive containing these files.

**-v :** verbose mode

**note : the find command is used to to find files the files**

**find <search directory> <search criteria>**

| * **-name** searches on the name of the file, * **-perm** search on the access rights of the file, * **-links** search on the number of links in the file, * **-user** searches for the owner of the file, * **-group** searches on the group to which the file belongs, | * **-type** searches on the type (**d = directory, c = character, f = normal file**), * **-size** searches on the size of the file in number of blocks (**1 block = 512 bytes**), * **-atime** search by date of last read access to the file, * **-mtime** search by last modification date of the file, * **-ctime** searches by file creation date. |
| --- | --- |

* **dd** : **dd if=<fichier(s)> of=<archive>**

**if=FILE :** read from FILE instead of stdin

**of=FILE :** write to FILE instead of stdout

**note : doesn’t save directories only files**

* **dump** : **dump -f <archive> <fichier(s)>**  : ext2/3/4 filesystem backup

**-f file :** Write the backup to file; file may be a special **device file like /dev/st0** **(a tape drive)**, **/dev/rsd1c (a floppy disk drive)**, **an ordinary file**, or **- (the standard output)**. **Multiple file names** may be given as **a single argument separated by commas**.

**note : dump allows you to backup a whole partition or a full directory. You can't use DUMP to backup a subdirectory, DUMP work only on partition.**

* **Restaure** :
* **tar** : **tar -xvf <archive> <fichier(s)>**

**-c :** creates an archive

**-v :** verbose mode

**-x --extract, --get :** Extract files from an archive. Arguments are optional. When given, they specify names of the archive members to be extracted.

**-t :** display the archive content

**-C <emp> :** where to extract the files

* **cpio** : **cpio -idvE tmp < <archive>**

**tmp** a file containing the file names to restaure

**-i, --extract :** Copy-in. Read the archive from standard input (or from the file supplied with the **--file option**) and extract files from it, or (if the -t option is given) list its contents to the standard output.

**-E, --pattern-file=FILE :** Read additional patterns specifying filenames to extract or list from FILE.

**-d, --make-directories :** Create leading directories where needed.

* **dd :**  **dd if=<emplacement> of=<archive>**

**note : juste swap the input and output file to restaure**

* **dump :** Comparer le contenu de l'archive avec l'original :

**to compare the archive content with the original :**

**restore -C -f <archive>**

**Interactive mode :**

**restore -i -f <archive>**

**--------------------------------------------------------------------**

**TP6 : Automate system administration tasks by**

**scheduling jobs to run in the future**

**--------------------------------------------------------------------**

**—-------- Commands used —--------**

* **scheduling jobs : crontab < [FILE]**

**crontab** : is a file containing the schedules which are lines having the following format :

**[Min] [H] [D] [Month] [DayWeek] [User] [Cmd]**

where :

**[Min] :** minute field **[0-59]** (**\***: **every** minute)

**[H] :** hour field **[0-23]** (**\***: **every** hour)

**[D] :** Day of the month **[1-31]** (**\***: **every** day )

**[Month] :** month field **[1-12]** (**\***: **every** month)

**[DayWeek] :** day of the week **[0-7]**

(**\***: **every** day and **0** refer to **sunday** )

**[User] :**  User field **(facultatif)**

**[Cmd]:** Command to execute (or specify the script’s path)

**FILE :** contains lines having this format

* **crontab -l :** lists the scheduled jobs

**—-------- Other Tips and commands —--------**

* printf "%s >> %s \n" "**${script}**" **"/tmp/Output" >> /tmp/fcron**;

-> prints a line with the value of the variable **script** and append it to the content of **/tmp/fcron** file | **/tmp/fcron** contains the lines to put in **crontab**

* printf "%s" "date = " ; date +%D ;

printf "%s" "heure = " ; date +%T ;

-> displays the date and hour of execution

**--------------------------------------------------------------------**

**TP7 : Configure and use system log files to meet**

**administrative and security needs**

**--------------------------------------------------------------------**

**—-------- Commands used —--------**

* **df (disk free) :** permet d’afficher la valeur d'espace disque disponible des systèmes de fichier dont l'utilisateur possède l'accès en lecture.

**-h :** human-readable

* **awk :** permet d'appliquer un certain nombre d'actions sur un fichier. La syntaxe est inspirée du C

**awk [-Fs] [-v variable] [-f fichier de commandes] 'program' fichier**

**[-Fs] :** spécifie les séparateurs de champs

**[-v var] :** Définie une variable utilisée à l'intérieur du programme.

**[-f fichier] :** Les commandes sont lues à partir d'un fichier.

* **sudo dmesg (driver message | display message )** : utilisée pour examiner le tampon en anneau du noyau et imprimer le tampon de message du noyau (**message buffer of kerne**l) . La sortie de cette commande contient les messages produits par les pilotes de périphérique.
* **find / -exec wc -c {}\; 2> /dev/null | sort -r -n | head -n $nb :** displays the $nb largest files
* **sudo rm $fichiers :** delete files

**—-------- Other Tips and commands —--------**

* **/var/log/syslog** and **/var/log/messages** store all global system activity data, including startup messages. **Debian-based** systems like **Ubuntu** store this in **/var/log/syslog**, while **Red Hat-based** systems like **RHEL** or **CentOS** use **/var/log/messages.**
* **df -h | awk '{ print $4 , 100-$5"%" , $6 }' :** Afficher l'espace du disque restant