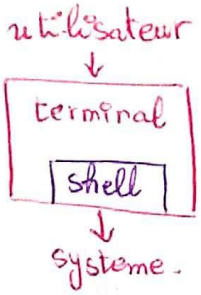
Systeme:

* Shell: l'interpteur des commandes.

utilisateur → shell → système.

commande → interpréte → éxécute

* terminal: programme qui éxécute des commandes.



* Utilisateur @ la machine : repertoire $.

≠ root $: self-utilisateur.

Commande

interne — externe

- Sont internes au shell
- sont éxecutes au sein de celui-ci.
- font partie du shell

- sont des programes externe.

Syntaxe: type - nom de la commande.

interne ⇒ help / history : est une primitive du shell.

externe ⇒ man : il nous donne le chemin pour arriver à la commande.

* Des commandes:

Date: commande sans option et sans paramétre.

date -d: sunday: commande avec option et paramétre.

date + %d %m %y : commande avec paramétre.

Commande	fonctionnement
echo "Hi"	Hi : écriture.
passwd	changer le mot de passe.
Killall systemd	fermer la session.
id -un	le nom d'utilisateur.
hostname -i	Ip adress
mkdir	créer un répertoire | mkdir -p dernier réel
rmdir	remove directory (répertoire vide).
rm -r	supprimer les répertoire vides et pleins
ls	liste (lister le contenu d'un répertoire).
cd	changer le repertoire actif.
mv	renommer ou déplacer des fichiers
rm	supprimer les fichiers
touch	créer des fichiers textes vides
cat, more, less	lire le contenu d'un fichier
which, where is	cherche des programmes.
cp	copier un fichier.
file	type de fichier.
grep	chercher un mot dans un fichier.
ls -r	afficher les répertoires cachés et non cachés.
hostname	affiche le nom de la machine
date	affiche la date
history	l'historique des commandes saisies
cal 06 2019	Calendrier de juin 2019
pwd	afficher le répertoire courant.
whoami	utilisateur.

history - c	supprimer l'historique
date %m %d %y	affiche le jour / mois / année
du	donne la taille d'un (fich / rep)
du - h	Donne la taille en format lisible
du - d	spécifier les prendon (niveau 0,1,2
tree	elle affiche les niveaux.
ln	un link / lien
ln - s	créer un fichier (raccourci)
ln - s	c'est un lien symbolique pour trouver les fichiers et les répertoire facilement. ln - s chemin absolu [fich / rep] chemin absolu d'arrivé / lien.
find / locate	pour chercher des fichiers ou des reper- find [repertoire] [expression] find Bureau/ "1.png"
Caractéres Spéciaux (*, ?) astérisque. ^ user tout qui commence par user	• chaine : suite de caractéro par ex: Bonjour / Allez / yousra. (*): chaine de taille variable (vide) (?): un caractére unique quelconque. ls /home/*/Bureau /home/admire/Bureau /home/mama/Bureau : ls /home/user?/Bureau /home/user01/Bureau. /home/user2/Bureau.
uname / uname-a	affiche des info sur le systeme d'exploitation.
~ : répertoire personel	/home/user01 (nom utilisateur)
. : répertoire courant .. : repertoire pere	• chemin absolu : commence par la racine / : /home/user1/Bureau. • chemin relatif : par rapport au répertoire courant (ex jesuis) Bureau/cours/tk.pdf

- Les droits d'accées:

- Linux est un systeme multi-utilisateur

✓ Lorsque administrateur créer un utilisateur un groupe sera automatiquement crée porte la même nom de l'utilisateur et contien que lui.

UID: user identification.

GID: Groupe identification.

quelque commande:

* groups: affiche les groupes.
* add user / add group: que le route peut les utiliser
* id - u: afficher le UID de l'utilisateur.
  id - g: afficher le GID des groupe
  id - G: afficher le tout (UID, GID).
* ls - l: afficher les droits d'accées d'un fichier
* ls - al: " " "
* ls - ld: afficher les droits d'un répertoire

(-) fichier / (d) répertoire / (ln / l): lien

r: lire: on peut lire le contenu.
w: écrire modifier le contenu du répertoire
✍ (ajouter, créer, supprimer des fichiers de répertoire).
x: accéde la commande cd / change directory éxecuté un programe.

* chmod: changer les droits d'accès.
* chmod -R: changer les droits de manière récursive.
* chgrp: changer le groupe d'un fichier ou d'un repertoire.
* chmod o+t: sticky bit (proteger le répertoire).

* sudo -s : passer en mode superutilisateur.

* chmod u+s : éxécuter la commande (programe avec le drôit de son propriétaire (bit SUID) ⇐

* Il donne a l'utilisateur le drôit du propriétaire.

* chmod g+s : bit GUID (même come SUID) (pour les groupes).

apt-get update : } mise à jour système
apt-get upgrade : } (root only).

apt-get install package : installer un package (root only).

* umask : affiche la valeur du masque.

* umask [valeur] : changer la valeur du masque.

r → 4
w → 2
x → 1
- → absence du drôit.

NB : le drôit de suppression n'est pas lié aux drôits du fichier, mais aux drôits du réperloire. (x) et le drôit w sur le repertoire

* le propriétaire peut changer le groupe d'un fich/rep ssi il fait partie du nouveau groupe.

+ : ajouter des drôits.
- : retirer des drôits.
= : effecter des drôits.
u : user. o : others.
g : group. a : user + group + other.

- Suid → 4
- Guid → 2
- Sticky bit → 1.

♡ avant d'installer un package, il faut mettre le système à jour

⇒ drôit automatique :
777 : d'un répertoire.
666 : d'un fichier.

chmod a=rwx ou chmod 777. | g=rw- u=rwx o=rw- ou chmod 766.

⇒ Filtrage :

♡ les commandes de trôitement des données :

* cat : afficher le contenu du fichier.

* cat -n ou nl : afficher le contenu du fichier avec numérotation des lignes.

* more : afficher le contenu du fichier sans revenir en arrière.

* less : afficher le contenu dans lequel on peux revenir en arrière.

* hexdump : affiche le contenu en hexadécimal.

* tac : affiche le contenu du fichier à l'envers.

la commande le fichier.

tr [A-Z] [a-z] : changer Maj → Min

sed -e s/$/"chaîne de charactères"/ fichier
mettre cette chaîne à la fin de chaque ligne du fichier.

La commande Sed :

pour remplacer une expression par une autre.

sed -i s/ancien expression/nouveau expression /g fichier.

La commande WC :

la commande (word count)

wc -l : compte le nombre de lignes.

wc -c : compte le nombre d'octets (caractère)

wc -w : compte le nombre de mots.

wc : show them all

ligne mots caractère.

La commande aspell :

interactive spell checker.

un correcteur orthographique interactive

aspell -c [check (vérifier l'erreur)]

-l : spécifier le langage

fichier.

for exemple :

user1@PC:~ $ aspell -l en US -c fichier

La commande split :

Découpage d'un fichier en corceaux.

split -b 2 [fichier] /split -l ↑ le nombre

La commande join :

to join two files

join fichier1 fichier2.

La commande diff / cmp :

pour comparer le contenu de deux fichers

⇒ Groupement des commandes, redirections :

* l'éxécution des commandes :

Chaque fois qu'on éxécute un commande un code de retour sera dans un variable. $?, on peut savoir sa valeur par l'éxécution de la commande echo $?

0 ⇒ si la commande est bien éxécutée.

1 ⇒ si il y a pas de résultat.

2 ⇒ si il y a une erreur (2 ou plus de 2)

* Groupement de commandes :

- **le point virgule (;) :**

pour éxécuter plusieurs commandes sur la même ligne.

user @ machine : ~/Bureau $ rm ; mv ; cp.

- **le pipe ou tube (|) :**

rediriger directement la sortie d'une commande vers l'entrée d'une autre commande.

user @ machine : ~ /Bureau $ cat fichier | sort -k 2

↑ on va trier le fichier

- **les charactères (& &)**

Commande 1 && Commande 2

commande 2 sera éxécutée uniquement si commande 1 à retourné 0 (reussite).

- **les deux pipes (||) :**

Commande 1 || Commande 2

commande 2 sera éxécutée uniquement si comande 1 à retourné un code different de 0 (n'a pas reussite / aucune résultat)

- Les canaux :

un canal : un fichier, qui possède son propre descripteur, et dans lequel on peut ou lire ou écrire.

① Canale d'entrée (stdin) => descripteur 0 comme le clavier

② Canale de sortie (stdout) => descripteur 1. comme l'écran.

③ Canal d'erreur (stderr) => descripteur 2. l'affichage des erreurs dans l'écran.

user1@PC:~$ pwd ; Hostname.
/home/user1 ← stdout.
HOStname : commande introuvable ← stderr.

1/ Redirection le canal de sortie standard vers un fichier : (stdout).

1> ou bien > : permette de rediriger la sortie de la commande vers un fichier et écrasé sa contenu puis le remplace par les nouvelles données.

user1@PC:~$ pwd > fichier
user1@PC:~$ cat fichier
/Rome/user1.

1>> ou bien >> : ~~rejet~~ rediriger la sortie de la commande vers un fichier sans écrasé sa contenu.

user1@PC:~$ cat fichier
Yousra
user1@PC:~$ pwd >> fichier
user1@PC:~$ cat fichier
Yousra
/home/user1

2/ Redirection de le canal d'erreur (Stderr) :

2> : pour rediriger avec écrasement.

2>> : pour rediriger sans écrasée le contenu de ce fichier.

Note : Si il y a pas de fichier, ce fichier sera crée avec cette redirection.

3/ Redirection de les deux canaux (stdout) (stderr) vers le même fichier :

> fichier 2>&1 : avec écrasement.

>> fichier 2>&1 : sans écrasement.

Exemple :
user1@PC:~$ {pwd; HOSTNAME} > fichier 2>&1
user1@PC:~$ cat fichier
/home/user1
HOSTNAME: commande introuvable.

4/ Redirection le canal d'entrée vers une commande : (stdin)

Commande < fichier

Exemple :
user1@PC:~$ wc < fichier
11 28 144

Commande << Chaîne de charactère (clavier)

user1@PC:~$ sort -n << END.
> 22
> 4
> 1
> 10
> END
> 1
> 4
> 10
> 22

Les variables

Les variables sont utilisé pour stocker temporairement une donnée (information).

✓ pour déclarer un variable :

nom_variable = valeur (sans éspace).

⇒ la valeur du variable est local dans le terminal, elle disparait quand le terminal est fermé.

par exemple: user1@PC:~$ a=yousra.
variable.

⇒ pour appeler un variable:

Exemple: user1@PC:~$ a=yousra
user1@PC:~$ echo $a

résultat ⇒ yousra

⇒ Si la variable d'un variable contien des espaces: on utilise (" ") : les guillemets

Exemple: a=" Bonjour Yousra"

echo $a: Bonjour Yousra.

Si on veux afficher l'espace donné entre deux variable: on utilise (" "):

par ex: a = yousra b = girl

echo "$a ⟷ $b"
yousra ⟷ girl

sinon echo $a ⟷ $b
yousra ⟷ girl

⇒ Pour ne pas interpréter la valeur d'un variables on utilise:

les apostrophes (' ')

par ex: a = Linux.

echo $a → Linux

echo '$a' → $a

Si notre valeur est une commande.

- On met (` `) pour l'éxécuté.
accent grave

exemple: a=`pwd`

echo $a → /home/user1

a=`date`

echo $a → samedi 01 février 2020:00 42

Les accolade { }: pour identifier le nom du variable:

Ex: a=fich.

echo ${a}1 ⇒ fich1.

cat ${a}1 ⇒ Hello world.

cat fich1 ⇒ Hello word.

+= : pour ajouter une valeur à un variable.

Ex: a=yousra.

a+="_amira" ← pour le petit espace.

echo $a → yousra_amira

la commande let: permet d'effectuer des opération mathématique:

ex: let a=5*10

echo $a → 50

Les variables système (prédéfinie):

pour les afficher "env".

$LOGNAME: user1.

$HOME: /home/user1

$SHELL: /bin/bash.

$HOSTNAME: PC.

$PWD: /home/user1

$USER: user1

exemple: user1@PC:~$ PWD=/home/Bureau
user1@PC:/home/Bureau $

$PATH: contient une liste des répertoires ou les commandes sont trouvé, séparé par ':'

Les processus:

un processus: un programme au cours d'éxécution.

les commandes:

ps: permet d'affiche les processus en cours d'éxécution lancé par l'utilisateur depuis la console actuelle.

ps:
- PID : Process ID, numéro du processus
- TTY, le nom du terminal depuis lequel le processus à été lancé.
- TIME : durée de traitement
- CMD : commande éxécuté

ps-f
- UID : nom d'utilisateur
- PPID : ID du processus père
- C : facteur de priorité
- STRIME : heure du lancement du processus.

ps-l
- PRI : priorité du processus
- NI : valeur entre "20" et "-20" : plus la valeur est élevée plus le traitement est ralenti

ps-u : affiche les processus d'un utilisateur ou plusieurs utilisateur séparé par (:).
(tous les processus)

Si le TTY est ? : alors le processus n'est pas associé à aucun terminal.

ps-e : que les processus en cours d'éxécution, et qui sont pas associé à aucun terminal.

nice [-valeur] commande [argument]
modifié la priorité d'un processus au démarrage du processus.

Ex: nice -10 sleep 100 &

PRI	NI		PRI	NI
80	0	<==>	90	10
80	0		80	0

renice [-n val] [-p PID] [-g GID] [-u UID]
pour modifier la priorité d'un processus aprés démarrage.

ex: renice -n 10 -p 4797

PID	PRI	NI
4797	90	10

pstree affiche les processus en cours d'éxécution sous forme d'un arbre.

pstree -p : selon les PID (afficher l'arbre qui contient les PID)

pstree -u : afficher l'arbre avec les noms d'utilisateur.

pstree -p | head -n 6

top: afficher les processus en temps réel.
avec un refreche du 3 s.

pgrep: [motif]
affiche le PID d'un processus rechercher.

ex: pgrep bash.
7422
7544

pgrep -bash -u user1
7422

Kill - N° du signal PID:
pour forcer l'arrêt d'un processus

Kill -9 : finir le process brutalement

Kill -15 : termine le processus normalement.

- Pour lancer le processus en arrière plan.
&.
↑ Ctrl + C pour le fermer.

Attention :

Seul root peut diminuer la valeur de priorité, chaque utilisateur peut juste augmenter sa valeur de priorité.

Priorité root هو الوحيد الذي يستطيع التقليل من أو الزيادة فيها.

المستعمل العادي يستطيع فقط التقليل منها.

* Chaque utilisateur peut augmenter sa valeur, ms le root peut augmente ou diminuer pour toute les utilisateurs.

* pstree
→ -p : pour afficher les PID.
→ -u : pour afficher les utilisateurs

rôle : affiche les processus en cours sous forme d'arbre.

top : permet d'afficher les informations des processus en temps réel, le refraichissement se fait à chaque 03 second.

pgrep : permet les utilisateurs de rechercher des processus à partir d'un nom ou autre attributs. ⇒ elle affiche le PID.

pgrep [options] [motif]

exemple :

```
user1@PC:~$ ps -ef | grep ^ user1.
user 1@PC:~$ pgrep bash
70242
74024
user 1@PC:~$ pgrep bash -u user 01
70242.
```

Kill : permet d'envoyer un signal à un processus. (gestionnaire des tâches).

Kill -l : permet d'afficher la liste des signaux.

Kill -9 PID : ferme le processus brutalement (force stop).

Kill -15 PID : il demande au processus de se terminer normalement. (fermé)

autant d'utiliser Kill, on peut clicker sur Ctrl + C.

Les Scripts

le nom du script.sh :

✓ Il faut le commencer par #!/bin/bash.

✓ les commentaire commencent par un #.

✓ Il faut donner le script le droit d'éxécution (car Il est un fichier qui contient des commandes à éxécuté).

chmod +x nom_duscript.

✓ Pour éxecuté ce script il faut le précédé par ./nom_script.

ou bien ajoutant le répertoire courant dans $PATH.

PATH = $PATH:. / PATH+=:.

* les paramètres:

$0: nom du script.

$1 ... $9: les paramétres nom_script par1 par2

$#: nombre de parametres.

$*: toute les paramètres comme une chaîne.

$@: toute les paramètres comme une liste.

* read variable variable2
echo $variable
echo $variable2

* read -p 'une chaîne :' variable
echo : une chaîne $ variable

-s: pour n'a pas afficher le variable comme les mots de passe.

echo -e \n: retour à la ligne.

le test [-option param] sur fichier

-f: Si c'est un fichier. [-f fichie]

-d: Si c'est un répertoire.

-e: Si le fichier ou le répertoire existe.

-s: Si " " " n'est pas vide.

-r: Si le user avait le droit de read

-x: " " " éxecute

-w: " " " write

-u: si le programme à le bit suide

-g: " " " guide

[-option fich/pro/rep]

Testa sur les chaînes de caractères:

= : les deux chaîne sont égaux.

!= : les deux chaîne sont différents.

-z ' ' : si la chaîne est vide.

-n 'chaîne': si la chaîne n'est vide.

Tester les valeur némuriques.

-eq : =	-ne ≠
-gt : >	-ge ≥
-lt : <	-le ≤

Test combiné:

-a: and / et logique.

-o: or / ou logique

-!: Not / non logique.

[-r file -o -w file] && "teste combinés"

[! -x Linux] && echo 'n'as pas le droit x'

La boucle If:

```
If [condition]
then
commande
else
commande
fi
```

Exemple:

```
#!/bin/bash
If [ $1 = $2 ]
then
echo " $1 égale a $2"
else
echo " $1 n'égal pas $2"
fi
user1@pc:~ $./script 10 15
10 n'égal pas à 15.
```

La boucle Case:

```
case [$variable] in
pattern1)
        Commande 01
    ;;
pattern2)
        commande 02
    ;;
esac
```

Exemple:

```
#!/bin/bash
read-p 'entrer votre age' : STRING
case $STRING in
dix )
    echo "you are so young"
    ;;
mille)
    echo" you ate dead"
;;
```


La boucle for:

```
For variable in liste
do
commande
done
```

Exemple:

```
user1@pc:~$ ls Bureau/
fich1 fich2 mama
user1@pc:~$ cat essai_for
#!/bin/bash.
for obj in Bureau/*
do
  mv $obj $1
done
user1@pc:~$ ./essai_for Document/
user1@pc:~$ ls Bureau/
user1@CC:~$ ls Document/
fich1 fich2 mama.
```

+ Exécution automatique:

at: permet de planifier des tâches à éxécuter automatiquement une seul fois.

```
at [options] [temps] [date].
at>
at>
at><EOT> ← Ctrl+D.
```

at -l: lister les tâches programé (les infos)

at: les commandes. (la mains).

at -f: pour automatisé les scripts.

```
at -f nom_script [le temps]
```

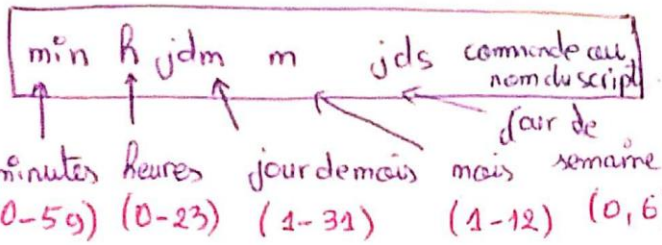
at -d: supprimer une tache /at -d /25 26

crontab: planifier les tâches à éxecuter régulièrment (modifier le fichier crontab).

-e : édite le fichier « crontab ».

-l : liste le contenu du fichier « crontab »

-r : supprime le contenu du fichier



* * * * * : tout les minutes (chaque minutes).

*/10 * * * * : chaque 10 min.

0 0 * * 1,3,4 (A minuit, le lundi mardi, jeudi).