

Sujet 103 : Commandes GNU et Unix

Présenté par : Mohamed Belhassen
2019–2020

Sujet 103 : Commandes GNU et Unix

- ▶ 103.1 Travailler en ligne de commande (**Weight 4**)
- ▶ 103.2 Contrôler des flux de texte à l'aide des filtres (**Weight 3**)
- ▶ 103.3 Effectuer la gestion de base des fichiers (**Weight 4**)
- ▶ 103.4 Utilisation des flux, des tubes (pipes) et des redirections (**Weight 5**)
- ▶ 103.5 Création, surveillance et destruction de processus (**Weight 5**)
- ▶ 103.6 Modifier la priorité d'exécution d'un processus (**Weight 3**)
- ▶ 103.7 Recherche sur des fichiers texte avec des expressions régulières (**Weight 2**)
- ▶ 103.8 Édition de fichiers texte avec "vi" (**Weight 3**)

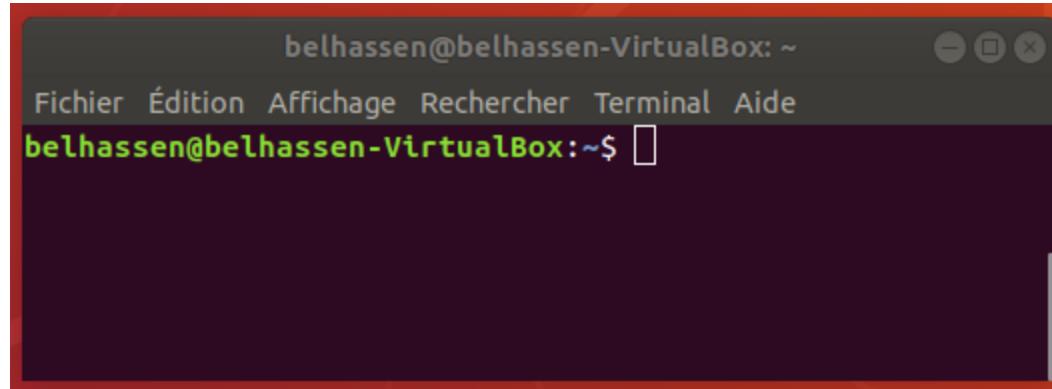
Travailler en ligne de commande

Travail en ligne de commande

- ▶ **Description:** Les candidats doivent être capables de travailler en ligne de commande.
l'utilisation du shell bash sera traité dans cet objectif.
- ▶ **Termes, fichiers et utilitaires utilisés :**
 - bash
 - echo
 - env
 - exec
 - export
 - pwd
 - set
 - unset
 - uname
 - history

bash

- ▶ Bourne–again shell compatible avec sh, avec des fonctionnalités de **ksh**, **csh**
- ▶ un shell est un programme qui exécute des programmes et permet aussi de construire d'autres programmes appelés scripts.
- ▶ **Prompt:**



A screenshot of a terminal window titled "belhassen@belhassen-VirtualBox: ~". The window has a dark background and light-colored text. At the top, there is a menu bar with options: Fichier, Édition, Affichage, Rechercher, Terminal, Aide. Below the menu bar, the title bar shows the user's name and host. The main area of the terminal shows a green prompt: "belhassen@belhassen-VirtualBox:~\$". There is a small square icon to the right of the prompt. The entire terminal window is framed by a red border.

Séquences de commandes

- ▶ Exécuter **séquentiellement** des commandes l'une après l'autre :
 - `cmd1 ; cmd2`
- ▶ Exécuter `cmd2` **si et seulement si** `cmd1` s'est exécutée **sans erreur**
 - `cmd1 && cmd2`
- ▶ Exécuter `cmd2` **si et seulement si** `cmd1` a renvoyé une **erreur** :
 - `cmd1 || cmd2`
- ▶ **&** en fin de commande permet de lancer cette commande en **tâche de fond** (background)
 - `firefox &`

Variables simples

```
$ formation="Lpi"
```

```
$ echo $formation
```

Lpi

- ▶ Rendre la variable visible pour les shells fils :

```
$ export formation
```

- ▶ Afficher toutes les variables d'environnements :

```
$ env
```

- ▶ Afficher les variables simples et les variables d'environnement :

```
$ set
```

- ▶ Effacer une variable :

```
$ unset formation
```

Quelques variables d'environnement bash

Variable	Fonction
USER	le nom de l'utilisateur courant
UID	UID de l'utilisateur courant
HOME	Le répertoire de connexion de l'utilisateur courant
?	Le code d'erreur de la dernière commande
#	Le nombre de paramètres à l'appel du script
1,2....	les paramètres du script
0	le nom du script

Exemple

- ▶ nano test.sh
- ▶ Mettre dedans:

```
#!/bin/bash
echo $1
echo $2
echo $#
```
- ▶ Rendre le fichier exécutable:
chmod +x test.sh
- ▶ Puis exécuter avec les appels suivants:
. ./test.sh formation LPI
. ./test.sh formation
. ./test.sh

quotes et variables

- ▶ **Quote double** : Permet la substitution des variables

```
$ echo "mon repertoire est $HOME"
```

mon repertoire de connexion est /home/belhassen

- ▶ **Quote simple** : Désactive l'interprétation des caractères spéciaux

```
$ echo 'mon repertoire de connexion est $HOME'
```

mon repertoire est \$HOME

- ▶ **Quotes inversées** : Permet la substitution des commandes

```
$ echo "mon repertoire courant est `pwd`"
```

mon repertoire courant est /tmp

Raccourcies claviers

- ▶ history
- ▶ HISTSIZE
- ▶ HISTFILE

Designator	Description
!!	Spoken as bang-bang, this command refers to the most recent command. The exclamation point is often called bang on Linux and Unix systems.
!n	Refer to command <i>n</i> from the history. Use the history command to display these numbers.
!-n	Refer to the current command minus <i>n</i> from the history.
!string	Refer to the most recent command starting with <i>string</i> .
?string	Refer to the most recent command containing <i>string</i> .
<i>string1</i> [^] <i>string2</i>	Quick substitution . Repeat the last command, replacing the first occurrence of <i>string1</i> with <i>string2</i> .

Prevent Recording Commands in History

- ▶ To prevent recording commands in the history list, temporarily disable recording by using:
 - `set +o history`
- ▶ To re-enable recording, use:
 - `set -o history`

Delete History

- ▶ Use the **-d** option with the **history** command to delete a command from the history list. For instance, delete command number 87 with:
 - `history -d 87`

Contrôler des flux de texte à l'aide des filtres

Traitement de flux de type texte par des filtres

- ▶ **Description:** Les candidats doivent être capables d'appliquer des filtres à un flux de type texte.
- ▶ **Termes, fichiers et utilitaires utilisés :**

cat
cut
expand
fmt
head
od
join
nl
paste

pr
sed
sort
split
tail
tr
unexpand
uniq
wc

Filtres (1)

Cut:
Extraire
des
champs

cat, less,
more:
Afficher le
contenu

sort: Trier
les lignes

nl:
Numéroter
les lignes

Fichier
Texte

tail / head:
Afficher
l'enête /
enqueue

wc: Afficher
le nombre
d'octets, de
mots et de
lignes

cat et tac

- ▶ Affiche le contenu d'un fichier.
- ▶ Exemple fichier1

1 un

2 deux

3 trois

- ▶ \$ cat fichier1

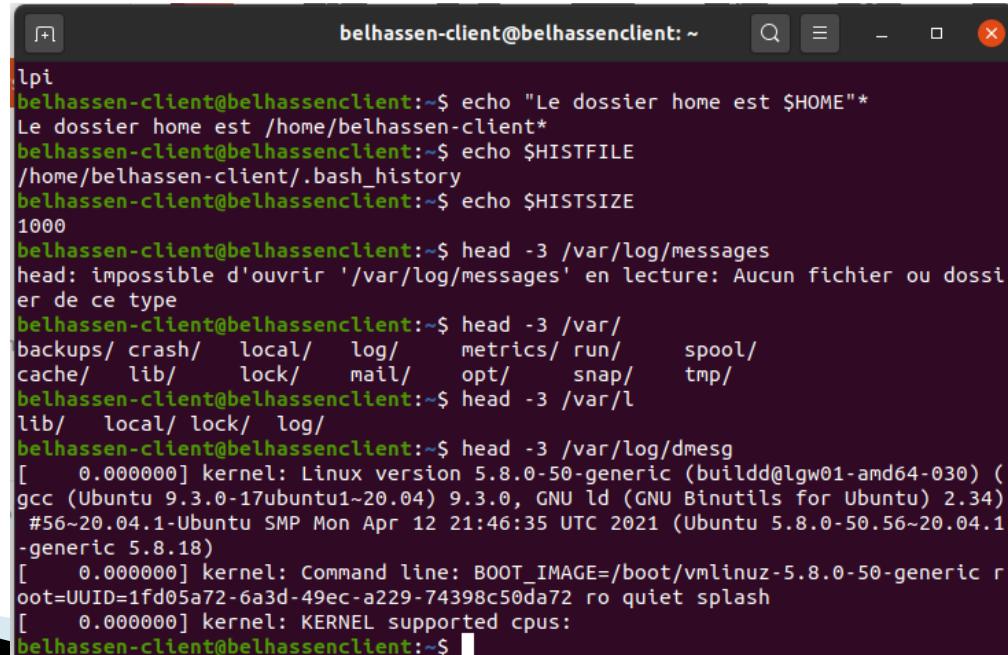
```
belhassen@belhassen-VirtualBox: ~
Fichier Édition Affichage Rechercher Terminal Aide
belhassen@belhassen-VirtualBox:~$ cat fichier1
1 un
2 deux
3 trois
belhassen@belhassen-VirtualBox:~$ □
```

```
belhassen@belhassen-VirtualBox: ~
Fichier Édition Affichage Rechercher Terminal Aide
belhassen@belhassen-VirtualBox:~$ tac fichier1
3 trois
2 deux
1 un
belhassen@belhassen-VirtualBox:~$
```

- ▶ \$ tac fichier1

head et tail

- ▶ **head** : Afficher le début d'un fichier (par défaut les 10 premières lignes)
- ▶ Exemple
- ▶ \$ head -3 /var/log/dmesg



```
lpi
belhassen-client@belhassenclient:~$ echo "Le dossier home est $HOME"
Le dossier home est /home/belhassen-client
belhassen-client@belhassenclient:~$ echo $HISTFILE
/home/belhassen-client/.bash_history
belhassen-client@belhassenclient:~$ echo $HISTSIZE
1000
belhassen-client@belhassenclient:~$ head -3 /var/log/messages
head: impossible d'ouvrir '/var/log/messages' en lecture: Aucun fichier ou dossier de ce type
belhassen-client@belhassenclient:~$ head -3 /var/
backups/ crash/ local/ log/ metrics/ run/ spool/
cache/ lib/ lock/ mail/ opt/ snap/ tmp/
belhassen-client@belhassenclient:~$ head -3 /var/l
lib/ local/ lock/ log/
belhassen-client@belhassenclient:~$ head -3 /var/log/dmesg
[    0.000000] kernel: Linux version 5.8.0-50-generic (buildd@lgw01-amd64-030) (
gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0, GNU ld (GNU Binutils for Ubuntu) 2.34)
#56~20.04.1-Ubuntu SMP Mon Apr 12 21:46:35 UTC 2021 (Ubuntu 5.8.0-50.56~20.04.1
-generic 5.8.18)
[    0.000000] kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-5.8.0-50-generic r
oot=UUID=1fd05a72-6a3d-49ec-a229-74398c50da72 ro quiet splash
[    0.000000] kernel: KERNEL supported cpus:
belhassen-client@belhassenclient:~$
```

head et tail

- ▶ **tail** : Afficher la dernière partie d'un fichier (par défaut les 10 dernières lignes)
- ▶ **Exemples :**
 - ▶ \$ tail -c20 /etc/passwd
 - Affiche les 20 derniers caractère du fichier passwd
 - ▶ \$ tail -f /var/log/syslog

Pour voir l'effet de cette commande: essayez par exemple (dans un autre terminal) d'installer / supprimer le paquet « rar »:
sudo apt-get install rar
sudo apt-get remove rar

cut

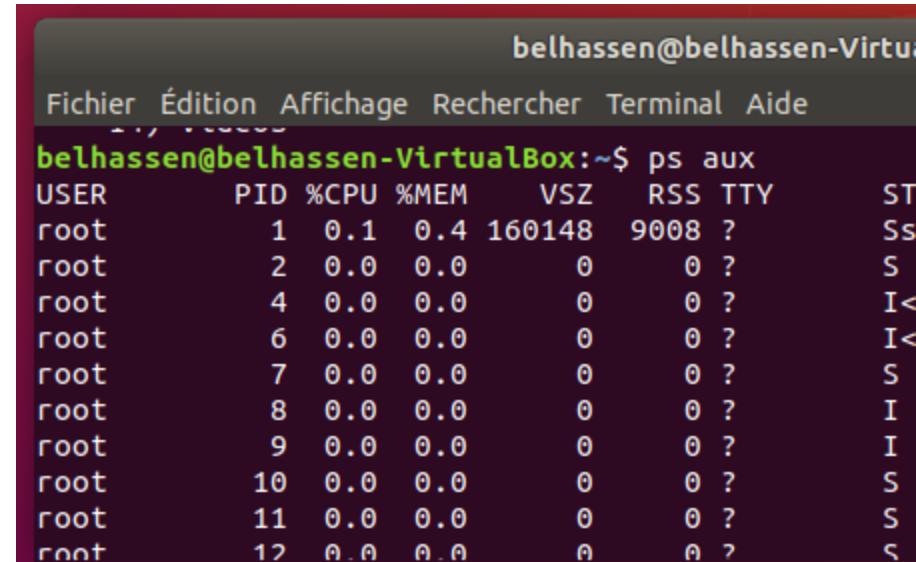
- ▶ Permet d'afficher certains champs d'un fichier donné
- ▶ Exemple :
- ▶ \$ cut -d: -f1 /etc/passwd

nl

- ▶ La commande **nl** permet de numéroter les lignes d'un fichier texte
- ▶ Exemples :
 - \$ nl /etc/passwd
 - \$ ls | nl -s')

sort

- ▶ La commande **sort** permet de trier les lignes d'un fichier texte
- ▶ Options
 - **-d** : Trier dans l'ordre alphabétique
 - **-n** : Trier dans l'ordre numérique
 - **-r** : Inverser l'ordre
- ▶ Exemples :
 - \$ sort /etc/passwd
 - \$ sort /etc/passwd -r
 - trier selon le champs RSS (resident size)
 - \$ ps aux | sort -k 6 -n
 - \$ ps aux | sort -k 6 -n -r



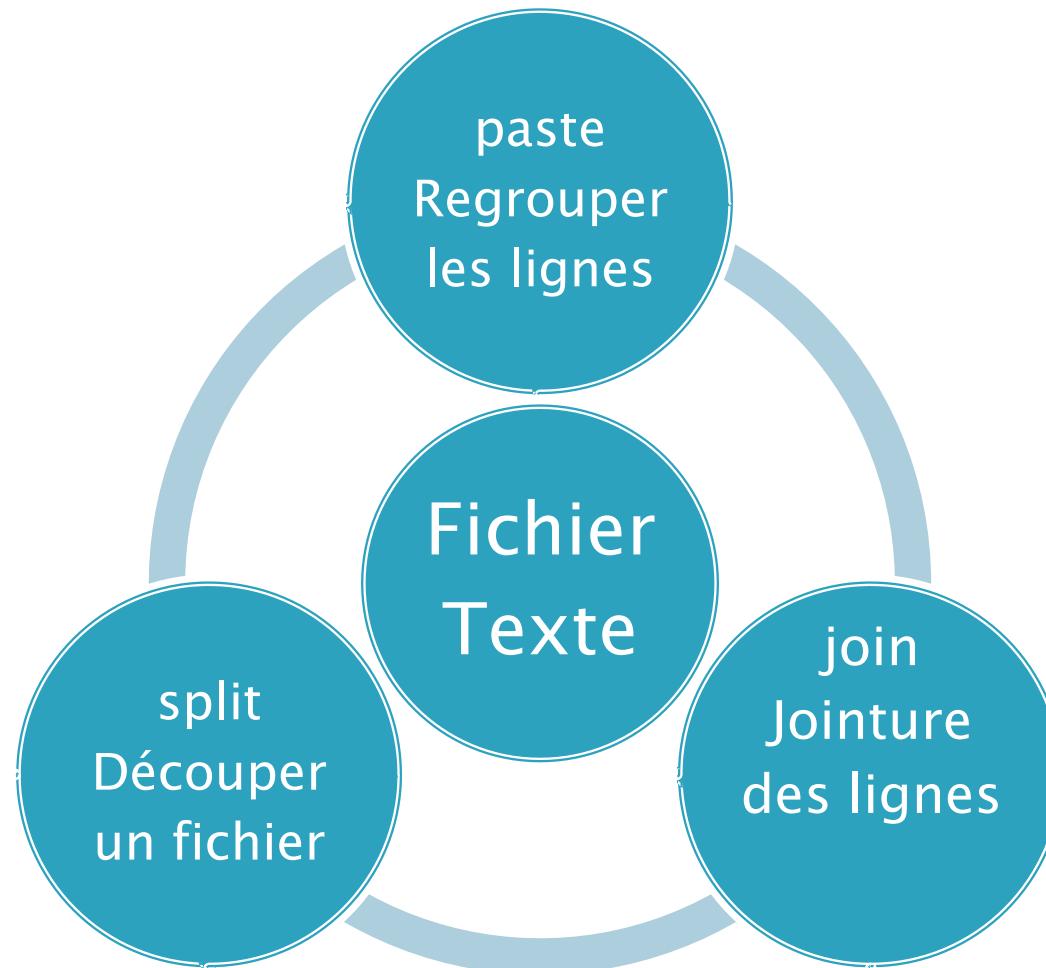
The screenshot shows a terminal window with a dark background and light-colored text. At the top, there's a menu bar with options: Fichier, Édition, Affichage, Rechercher, Terminal, and Aide. Below the menu, the prompt is `belhassen@belhassen-VirtualBox:~$`. The command `ps aux` is run, and its output is displayed. The output shows a list of processes with columns for USER, PID, %CPU, %MEM, VSZ, RSS, TTY, and ST. The processes listed are all 'root' processes, with PIDs ranging from 1 to 12. The RSS column values are 160148, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, and 0 respectively. The terminal window has a red border.

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	ST
root	1	0.1	0.4	160148	9008	?	Ss
root	2	0.0	0.0	0	0	?	S
root	4	0.0	0.0	0	0	?	I<
root	6	0.0	0.0	0	0	?	I<
root	7	0.0	0.0	0	0	?	S
root	8	0.0	0.0	0	0	?	I
root	9	0.0	0.0	0	0	?	I
root	10	0.0	0.0	0	0	?	S
root	11	0.0	0.0	0	0	?	S
root	12	0.0	0.0	0	0	?	S

WC

- ▶ La commande **wc** affiche le nombre de lignes, de mots et d'octets d'un fichier
- ▶ **Options:**
 - -c : Afficher uniquement le nombre d'octets
 - -m : Afficher uniquement le nombre de caractères
 - -l : Afficher uniquement le nombre de lignes
 - -w : Affiche uniquement le nombre de mots
- ▶ **Exemples :**
 - \$ wc -l fich
 - \$ wc -w fich
 - \$wc fich

Filtres (2)



paste

- ▶ La commande paste permet de regrouper les lignes de différents fichiers
- ▶ Exemple :

1	A
2	B
3	C

File 1 File 2

```
$ paste file1 file2
```

1	A
2	B
3	C

Resultat

paste (suite)

- ▶ \$ paste -d'@' file1 file2

```
1@A  
2@B  
3@C
```

Resultat

- ▶ \$ paste -s file1 file2

1	2	3
A	B	C

Resultat

join

- ▶ Fusionner les lignes de deux fichiers ayant un champ commun
- ▶ Exemple :

1	one
2	two
3	three

File 1

1	11
2	22
3	33

File 2

```
$ join -j 1 file1 file2
```

1	one	11
2	two	22
3	Three	33

Resultat

Join: cas de fichier non trié selon le champs de jointure

```
belhassen@DESKTOP-RPDNQFG: ~  
belhassen@DESKTOP-RPDNQFG:~$ cat f1  
1 un  
2 deux  
3 trois  
4 quatre  
belhassen@DESKTOP-RPDNQFG:~$ cat f2  
1 ali  
3 salem  
2 Nawal  
belhassen@DESKTOP-RPDNQFG:~$ join -j 1 f1 f2  
1 un ali  
join: f2:3: is not sorted: 2      Nawal  
3 trois salem  
belhassen@DESKTOP-RPDNQFG:~$
```

split

- ▶ Cette commande permet de **découper** un fichier en différentes parties
- ▶ Exemple 1:

```
1 one
2 two
3 three
4 Four
5 Five
6 Six
```

File 1

- ▶ `split -2 file1 splitout_`
==> créer trois fichiers `splitout_aa`, `splitout_ab`, et `splitout_ac`

split

- ▶ Exemple 2:
 - Soit un fichier image « PNG » de taille inférieure à 1 MB
 - `split -b 100k image.PNG splitted_image_`

==> créer des fichiers `splitout_aa`, `splitout_ab`, et `splitout_ac`, etc. de taille 100k (le dernier fragment sera $<= 100k$)

Restitution des fragments de fichiers créés avec « split »

 belhassen@DESKTOP-RPDNQFG: ~

```
belhassen@DESKTOP-RPDNQFG:~$ ls -l *.jpg
-rwxr-xr-x 1 belhassen belhassen 54320 Dec  9 08:53 zorro.jpg
belhassen@DESKTOP-RPDNQFG:~$ split -b 30k zorro.jpg splitted_zorro_
belhassen@DESKTOP-RPDNQFG:~$ ls -l splitted_zorro_*
-rw-r--r-- 1 belhassen belhassen 30720 Dec  9 09:00 splitted_zorro_aa
-rw-r--r-- 1 belhassen belhassen 23600 Dec  9 09:00 splitted_zorro_ab
belhassen@DESKTOP-RPDNQFG:~$ cat splitted_zorro_* > newZorroImage.jpg
```

Filtres (3)

Fmt: Formater les paragraphes

Od: Afficher en format octal

**Fichier
Texte**

Expand/
unexpand:
Convertir tabulations en espaces

Tr: Convertir des caractères

Pr: Mise en forme pour l'impression

fmt

- ▶ Cette commande permet de **formater** les paragraphes dans un fichier
- ▶ options :
 - **-u** : Espacement uniforme. Réduire les espacements entre les mots à une espace
 - **-w** : Remplir les lignes jusqu'à la largeur mentionnée (par défaut 75 colonnes)
- ▶ **Exemple 1:**
 - `$ fmt -w 80 myfile.txt > myfile80wide.txt`

Générateur de texte
aléatoire:
<https://fr.lipsum.com/>

fmt

- ▶ Cette commande permet de **formater** les paragraphes dans un fichier
- ▶ options :
 - **-u** : Espacement uniforme. Réduire les espacements entre les mots à une espace
 - **-w** : Remplir les lignes jusqu'à la largeur mentionnée (par défaut 75 colonnes)
- ▶ **Exemple 2:**
 - Créer un fichier contenant 2 paragraphes:
 - le premier contient un seul espace entre les mots
 - Le 2eme contient plus qu'un espace entre les mots
 - \$ fmt -u myfile.txt

Expand/unexpand

- ▶ La commande **expand** permet de convertir les tabulations d'un fichier en espaces
- ▶ **unexpand** fait le processus inverse
- ▶ Exemple :
 - convertir les tabulations en 2 espaces:
 - Soit un fichier « monfichier » contenant des tabulations
 - \$ expand -t 2 monfichier
 - Sauvegarder le résultat de unexpand:
 - \$ expand -t 2 monfichier>resultat
 - Afficher les tabulations qui existent dans un fichier
 - \$ cat -t monFichier

od

- ▶ Afficher le contenu d'un fichier en octal ou sous d'autres formats (decimal, hexadecimal, ASCII)
- ▶ option :
 - -t type Sélectionner le format d'affichage des résultats selon le type :
 - a : caractères littéraux
 - c : caractères ASCII ou séquences d'échappement préfixées par BackSlash
 - o : valeurs octales
 - u : valeurs décimales non signées
 - x : valeurs hexadécimales

od

- ▶ Afficher le contenu d'un fichier en octal ou sous d'autres formats (decimal, hexadecimal, ASCII)
- ▶ Exemples : soit un fichier texte file1
 - \$ od -t a file1
 - \$od -t c file1
 - \$ od -t x1 file1

pr

- ▶ Dans Linux/Unix, la commande pr est utilisée pour préparer un fichier à imprimer en ajoutant des **pieds de page**, des **en-têtes** et le texte formaté d'une manière appropriée.
- ▶ La commande pr ajoute en fait 5 lignes de marge en haut et en bas de la page.
- ▶ La partie d'en-tête affiche la **date** et l'**heure** de la dernière **modification** du fichier avec le **nom du fichier** et le **numéro de page**.

Pr: exemple 1

- ▶ Exemple 1 :
- ▶ Pour imprimer k nombre de colonnes, nous utilisons -k.
- ▶ Disons que nous avons un fichier qui contient 10 chiffres de 1 à 10 avec chaque numéro dans une nouvelle ligne.

```
akash@akash-Vostro-5568:~$ cat abc.txt
1
2
3
4
5
6
7
8
9
10
```

Pr: exemple 1

- ▶ Maintenant, si nous voulons imprimer ce contenu en 3 colonnes, nous utiliserons la commande suivante.
 - pr -3 abc.txt
 - ici abc.txt est le nom du fichier.

```
akash@akash-Vostro-5568:~$ pr -3 abc.txt  
2020-09-11 06:01          abc.txt          Page 1
```

1	5	8
2	6	9
3	7	10
4		

Pr: exemple 2

- ▶ Pour supprimer les en-têtes et les pieds de page, l'option –t est utilisée.
 - pr –t abc.txt
- ▶ Après avoir exécuté la commande ci-dessus, il nous donnera la sortie suivante.

```
akash@akash-Vostro-5568:~$ pr -t abc.txt
1
2
3
4
5
6
7
8
9
10
```

Pr: exemple 3

- ▶ Pour Doubler l'entrée de pas et réduire l'encombrement on utilise l'option -d:
 - pr -d abc.txt
 - Après avoir exécuté la commande ci-dessus, il nous donnera la sortie suivante.

```
akash@akash-Vostro-5568:~$ pr -d abc.txt

2020-09-11 06:01          abc.txt          Page 1

1
2
3
4
5
6
7
```

Pr: exemple 4

- ▶ Pour numérotter les lignes ce qui aide au débogage, l'option –n est utilisée.
 - pr –n abc.txt
 - Après avoir exécuté la commande ci-dessus, il nous donnera la sortie suivante.

```
akash@akash-Vostro-5568:~$ pr -n abc.txt

2020-09-11 06:01                      abc.txt          Page 1

      1   1
      2   2
      3   3
      4   4
      5   5
      6   6
      7   7
      8   8
      9   9
     10  10
```

Pr: exemple 5

- ▶ Pour remplacer le nom de fichier dans l'en-tête de chaque page par un texte personnalisé, on utilise la commande suivante:
 - ▶ **pr -h Texte abc.txt**
 - ▶ Au lieu d'afficher le nom du fichier, on peut écrire un texte personnalisé dans le titre

```
belhassen-client@belhassenc:~$ pr -h Texte pr_in  
a  
2021-11-25 09:11          Texte          Page 1  
b  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

tr

- ▶ Pour effectuer des **conversions** de caractères (exp **minuscule/majuscule**, ...)
- ▶ Exemples :
 - Convertir les minuscules en majuscules:
 - \$ cat file1 | tr 'a-z' 'A-Z'
 - \$ tr 'a-z' 'A-Z' < file1
 - OU \$ cat file1 | tr '[:lower:]' '[:upper:]'
 - la suppression des accents d'un texte :
 - cat file1 | tr 'àçéèêëñôöùüÂÇÉÈËÏÔÖÙÜ'
'aceeeeiiouuuACEEEEIIOOUU'
 - Convertir les séquences de sauts de lignes en un seul saut de ligne (ceci supprime les lignes blanches) :
 - cat file1 | tr -s '\n'

Gestion de fichiers

Effectuer une gestion de base sur les fichiers

- ▶ **Description :** Les candidats doivent être capables d'utiliser les commandes Linux de base pour gérer les **fichiers** et les **répertoires**
- ▶ **Termes, fichiers et utilitaires utilisés :**
 - cp tar
 - find cpio
 - mkdir dd
 - mv file
 - ls gzip
 - rm gunzip
 - rmdir bzip2
 - touch

Inodes

- ▶ Associer à chaque objet du système de fichier un **inode** (The identification information for filesystem object)
- ▶ Un inode **regroupe** des **informations** sur **l'objet** du système de fichiers : localisation, date de modification, paramètres de sécurité....
- ▶ Chaque système de fichier **ext2** est créé avec un **nombre fini** d'inodes calculé selon la taille du système de fichier et d'autres options passées à la commande `mke2fs`
- ▶ plusieurs objets du système de fichiers peuvent partager le même inode: lien physique

Commandes de base

- ▶ **pwd** : Afficher le chemin absolu du répertoire courant
- ▶ **cd** : changer de répertoire
- ▶ **ls** : lister le contenu d'un répertoire
- ▶ **mkdir** : créer un nouveau répertoire
- ▶ **rmdir** : supprimer un répertoire
- ▶ **touch** : changer les informations de date et de d'heure d'un fichier; créer un fichier vide lorsque le fichier passé en argument n'existe pas.
- ▶ **cp** : copier un fichier
- ▶ **mv** : Déplacer ou renommer un fichier
- ▶ **rm** : supprimer un fichier

Utiliser les wildcards

- ▶ Besoin de manipuler plusieurs fichiers à la fois
- ▶ Par exemple: cas d'un développement en C et pour être sûre de recompiler tous les fichiers sources

Utiliser les wildcards

Wildcards	Descriptions	Exemples
*	désigne 0 ou plusieurs caractères	x*
?	Désigne exactement un seul caractère	x? x??
[caractères]	Désigne un seul caractère de la liste [caractères]	x[yz]
[!caractères]	Désigne un seul caractère en dehors de la liste [caractères]	x[!yz]
[a-z]	Désigne un seul caractère appartenant à l'intervalle de caractères défini entre []	x[0-9] x[a-zA-Z]
[!a-z]	Désigne un seul caractère n'appartenant pas à l'intervalle de caractères défini entre []	[!a-c]
{frag1,frag2,...}	brace expansion	file_{one,two,three}

Les tubes et les redirections

Utilisation des flux, des tubes (pipes) et des redirections

- ▶ **Description :** Les candidats doivent être capables de **rediriger** des flux et de les **connecter** dans le but de **traiter efficacement** ces **données textuelles**. Les tâches à effectuer comprennent
 - les redirections de **l'entrée standard**, de la **sortie standard** et de la **sortie standard des erreurs**,
 - connecter la sortie d'une commande à l'entrée d'une autre,
 - utiliser la sortie d'une commande comme paramètre pour une autre commande
 - et envoyer le résultat en même temps sur la sortie standard et dans un fichier.
- ▶ Termes, fichiers et utilitaires utilisés :
 - tee
 - xargs

Les tubes

- ▶ Les tubes Unix permettent de **combiner** des **commandes** en les utilisant comme des briques indépendamment de leur provenance
- ▶ Trois types d'entrées / sorties
 - Entrée standard (**stdin**) : Descripteur **0**
 - Sortie standard (**stdout**) : Descripteur **1**
 - Sortie d'erreur standard (**stderr**) : Descripteur **2**

Redirection

Fonction de redirection	Syntaxe
Envoyer stdout vers file	\$ cmd > file ou \$ cmd 1> file
Envoyer stderr vers file	\$ cmd 2> file
Envoyer stdout et stderr vers file	\$ cmd > file 2>&1
Envoyer stdout vers file1 et stderr vers file 2	\$ cmd > file1 2>file2
Recevoir stdin à partie de file	\$cmd < file
Ajouter stdout à la fin du file	\$ cmd >> file ou \$ cmd 1>> file
Envoyer stderr à la fin du file	\$ cmd 2>> file
Envoyer stdout et stderr à la fin du file	\$ cmd >> file 2>&1

Les tubes (pipe)

- ▶ La **sortie d'une commande** devient **l'entrée d'une autre**.
- ▶ **Tubes et redirections** peuvent être **combinées** sur une ligne de commande selon les résultats qu'on veut obtenir.
- ▶ Pipe stdout de cmd1 vers cmd2 :
 - `$ cmd1 | cmd2`
- ▶ Pipe stdout et stderr de cmd1 vers cmd2.
 - `$ cmd1 2>&1 | cmd2`
- ▶ Afficher les 6 premières lignes du fichier /etc/passwd une fois ce fichier trié par ordre alphabétique
 - `$ sort /etc/passwd | head -6`
- ▶ La commande **tee** permet de dupliquer le flux de données en sortie :
 - `$ sort /etc/passwd | tee res1.txt | head -6`

La commande xargs

- ▶ **cmd1 | xargs cmd2**
 - Permet de passer en arguments de la commande cmd2, le résultat de la commande cmd1
- ▶ **Exemple 1:** créer les nom de dossiers passés
 - **echo 'one two three' | xargs mkdir**
 - **ls**

one two three → Nouveaux dossiers créés
- ▶ **Exemple 2:** Supprimer les dossiers passés en paramètre de echo:
 - **echo 'one two three' | xargs -t rm -r**
 - L'option -t permet d'afficher la commande ainsi exécutée

Les processus

Création, surveillance et destruction de processus

- ▶ **Description :** Les candidats doivent être capables d'effectuer une gestion de base sur les processus.
- ▶ **Termes, fichiers et utilitaires utilisés :**
 - &
 - bg
 - fg
 - jobs
 - kill
 - nohup
 - ps
 - top
 - free
 - uptime
 - killall

Processus Unix ?

- ▶ Un processus est un **programme en cours d'exécution** qui utilise les **ressources**: de la mémoire + processeur.
- ▶ **Quelques informations relatives à un processus :**
 - PID : Process ID
 - PPID : Parent Process ID
 - User ID (UID) et Group ID (GID) : Ayant lancé le processus
 - temps CPU
 - tables de référence des fichiers ouverts

ps

- ▶ Quels sont les processus exécutés par le système?
- ▶ Afficher tous les processus du système :
 - # **ps -A**
 - ou **ps -ef**
- ▶ Manipulations
 - l'utilisateur **belhassen** exécute la commande :
 - \$ vi test
 - Afficher les processus de l'utilisateur **belhassen**
 - # ps -U **belhassen**
 - Afficher les utilisateurs qui exécutent la commande vi
 - # ps -f -C vi
 - UID PID PPID C STIME TTY TIME CMD
 - **1000** 5229 5201 0 18:23 pts/5 00:00:00 vi test

1000 : c'est l'identifiant de l'utilisateur belhassen:
cat /etc/passwd | grep '1000'

top

- ▶ Afficher des informations sur l'activité du système en temps réel

```
3:37pm up 46 days, 5:11, 2 users, load average: 0.01, 0.17, 0.19
96 processes: 94 sleeping, 1 running, 0 zombie, 1 stopped
CPU states: 0.1% user, 1.0% system, 0.0% nice, 0.9% idle
Mem: 1030268K av, 933956K used, 96312K free, 0K shrd, 119428K buff
Swap: 1052216K av, 1176K used, 1051040K free, 355156K cached

 PID USER      PRI  NI   SIZE   RSS   SHARE STAT %CPU %MEM   TIME COMMAND
22069 ian      17   0 1104 1104    848 R     0.9  0.1  0:00 top
  1 root      8   0  500  480   444 S     0.0  0.0  0:04 init
  2 root      9   0    0    0    0 SW    0.0  0.0  0:00 keventd
  3 root      9   0    0    0    0 SW    0.0  0.0  0:00 kapmd
  4 root     19  19    0    0    0 SWN   0.0  0.0  0:00 ksoftirqd_CPU0
  5 root      9   0    0    0    0 SW    0.0  0.0  0:00 kswapd
```

- ▶ Quelques options interactives :
 - h : help
 - n : nombre de processus à afficher
 - q : quitter
 - r : (renice) changer la priorité d'un processus

top

▶ Application:

- Tapez Z, puis la touche entrée → colorer l'affichage
- Tapez r
 - → Changer la priorité du processus de la commande top à 3
 - → Essayez de changer la priorité du processus de la commande top à -3 ou 1
- Tapez n, puis 5 → afficher seulement 5 processus

Envoyer un signal à un processus

- ▶ kill [numéro-du-signal] PID
- ▶ Afficher une liste des noms de signaux connus :
 - **kill -l**

Signal name ^[a]	Number	Meaning and use
HUP	1	Hang up. This signal is sent automatically when you log out or disconnect a modem. It is also used by many daemons to cause the configuration file to be reread.
INT	2	Interrupt; stop running. This signal is sent when you type Ctrl-C.
KILL	9	Kill; stop unconditionally and immediately. Sending this signal is a drastic measure, as it cannot be ignored by the process. This is the "emergency kill" signal.
TERM	15	Terminate, nicely if possible. This signal is used to ask a process to exit gracefully.
TSTP	20	Stop executing, ready to continue. This signal is sent when you type Ctrl-Z.
CONT	18	Continue execution. This signal is sent to start a process stopped by SIGTSTP or SIGSTOP. (The shell sends this signal when you use the fg or bg commands after stopping a process with Ctrl-Z.)

Envoyer un signal à un processus

► Exemple 1:

- Dans un premier terminal, tapez :
 - **\$ gedit fichier1**
- Dans un 2^{ème} terminal, chercher l'id du processus gedit:
 - **\$ ps -f -C gedit**
- Puis exéutez la commande suivante:
 - **\$ kill -9 idGeditTrouvé**

► Autres exemples:

- <https://www.geeksforgeeks.org/kill-command-in-linux-with-examples/>

Envoyer un signal à un processus

► Exemple 2:

- Dans un premier terminal, tapez :
 - \$ **gedit fichier1**
 - Puis, dans le même terminal, tapez **CTRL+Z**
- **Que remarquez vous pour la fenêtre d'édition de Gedit?**
 - Tapez maintenant, dans le même terminal:
 - **fg 1**

Envoyer un signal à un processus

- ▶ Envoyer SIGTERM aux processus (PIDs 1000 et 1001 (changer les numéros selon vos ID de processus)
 - \$ kill 1000 1001
 - \$ kill -15 1000 1001
 - \$ kill -SIGTERM 1000 1001
 - \$ kill -TERM 1000 1001
- ▶ relecture des fichier de configurations
 - kill -HUP `cat /var/run/httpd.pid`
→ Cette commande nécessite que le service httpd soit installé (serveur web)
- ▶ Arrêt forcé !
 - kill -9 1000 1001 ou bien kill -KILL 1000 1001

Envoyer un signal à un processus

- ▶ Afficher les processus qui s'exécutent en arrière plan (bg)
 - `# firefox &`
 - `[1] 5788`
 - `# jobs`
 - `[1]+ Running ./firefox &`

Envoyer un signal à un processus

- ▶ Vous avez oublié de lancer firefox en arrière plan (bg):
 - # **firefox**
 - (vous faites ctrl z) → TSTP (20)
 - [1]+ Stopped firefox
 - # **bg** → CONT (18)
 - [1]+ firefox &
- ▶ Envoyer un signal à des processus indiqués par leurs noms
 - \$ **killall firefox**

Modifier la priorité d'un processus

Modification des priorités des processus

- ▶ **Description :** Les candidats doivent être capables de gérer les priorités des processus.
- ▶ **Termes, fichiers et utilitaires utilisés :**
 - nice
 - ps
 - renice
 - top

Priorité des processus

- ▶ **top** ou bien **ps -l**
- ▶ le noyau offre + temps CPU pour « **high priority process** »
- ▶ Par défaut les processus d'un utilisateur sont créés avec la **Nice Number 0**.
- ▶ Nice Number positif --> moins de priorité
- ▶ Nice Number négatif --> plus de priorité
- ▶ Nice Number varie de **-20 à 19**
- ▶ Un utilisateur **peut lancer** un processus avec un **Nice Number positif**
- ▶ **SEUL root** peut lancer un processus avec un **Nice Number négatif**

Nice et renice

- ▶ Un utilisateur lance cmd avec un Nice Number +5
 - \$nice -5 cmd1
- ▶ Seul root peut lancer des processus avec un « Nice Number » négatif
 - # nice --10 vi /etc/hosts.deny
 - # nice -n -10 vi /etc/hosts.deny
- ▶ renice : Modifier la priorité d'un processus
 - #renice -20 501
 - #renice -10 -u belhassen -p 501
 - NB: changer « belhassen » par votre nom d'utilisateur et « 501 » par l'id de votre processus à modifier

Recherche sur des fichiers texte avec des expressions régulières

Recherche dans des fichiers texte avec les expressions régulières

- ▶ **Description :** Les candidats doivent être capables de d'effectuer des recherches sur le contenu des fichiers selon un modèle
- ▶ **Termes, fichiers et utilitaires utilisés :**
 - grep
 - egrep
 - fgrep
 - sed

Objectifs et outils

- ▶ Recherche (texte) sur le contenu des fichiers, selon un modèle (motif) : « les expressions régulières » **regex**
- ▶ Une expression régulières (regular expression) est un motif qui permet de décrire un ensemble de chaînes
- ▶ **Outils** : grep, egrep, sed, awk, Perl, java ..

grep

- ▶ grep [options] regex [fichiers]
- ▶ options :
 - **-c** : Afficher le nombre de lignes qui satisfait regex, pas les lignes
 - **-h** : Ne pas afficher le nom des fichiers dans les résultats lorsque plusieurs fichiers sont parcourus.
 - **-i** : Ignorer les différences majuscules/minuscules dans la recherche.
 - **-n** : Ajouter à chaque ligne de sortie un préfixe contenant son numéro dans le fichier
 - **-v** : Afficher les lignes qui ne satisfait pas regex
 - **-E** : Interpréter regex comme une expression régulière étendu. **egrep**

Tutoriaux sur grep

- ▶ <https://www.digitalocean.com/community/tutorials/using-grep-regular-expressions-to-search-for-text-patterns-in-linux-fr>
- ▶ <https://www.opensourceforu.com/2012/06/beginners-guide-gnu-grep-basics/>
- ▶ <https://ostechnix.com/the-grep-command-tutorial-with-examples-for-beginners/>
- ▶ <https://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/>
- ▶ <https://www.hostinger.com/tutorials/grep-command-in-linux-useful-examples/>

Expression régulière : position

Regular expression	Description
^	Match at the beginning of a line. This interpretation makes sense only when the ^ character is at the left-hand side of the <i>regex</i> .
\$	Match at the end of a line. This interpretation makes sense only when the \$ character is at the right-hand side of the <i>regex</i> .
\<\>	Match word boundaries. Word boundaries are defined as whitespace, the start of line, the end of line, or punctuation marks. The backslashes are required and enable this interpretation of < and >.

Expression régulière : groupe de caractères

Regular expression	Description
[abc] [a-z]	Single-character groups and ranges. In the first form, match any single character from among the enclosed characters a, b, or c. In the second form, match any single character from among the range of characters bounded by a and z (POSIX character classes can also be used, so [a-z] can be replaced with [:lower:]). The brackets are for grouping only and are not matched themselves.
[^abc] [^a-z]	Inverse match. Match any single character not among the enclosed characters a, b, and c or in the range a-z. Be careful not to confuse this inversion with the anchor character ^, described earlier.
.	Match any single character except a newline.

Expression régulière : Les modificateurs

Basic regular expression	Extended regular expression (egrep)	Description
*	*	Match an unknown number (zero or more) of the single character (or single-character <i>regex</i>) that precedes it.
\?	?	Match zero or one instance of the preceding <i>regex</i> .
\+	+	Match one or more instances of the preceding <i>regex</i> .
\{n, m\}	{n, m}	Match a range of occurrences of the single character or <i>regex</i> that precedes this construct. \{n\} matches n occurrences, \{n,\} matches at least n occurrences, and \{n, m\} matches any number of occurrences from n to m, inclusively.
\		Alternation. Match either the <i>regex</i> specified before or after the vertical bar.
\(regex\)	(<i>regex</i>)	Grouping. Matches <i>regex</i> , but it can be modified as a whole and used in back-references. (\1 expands to the contents of the first \(\) and so on up to \9.)

sed

- ▶ https://linuxhint.com/50_sed_command_examples/
- ▶ <https://www.geeksforgeeks.org/sed-command-in-linux-unix-with-examples/>
- ▶ <https://linuxconfig.org/learning-linux-commands-sed>
- ▶ <https://predictivehacks.com/unix-sed-command-tutorial-with-examples/>
- ▶ <https://www.tutorialspoint.com/sed/index.htm>

L'édition de fichiers textes avec VI

- ▶ <https://linux.goffinet.org/administration/traitement-du-texte/editeur-de-texte-vi/>
- ▶ https://www.youtube.com/watch?v=S24LN5h_pac