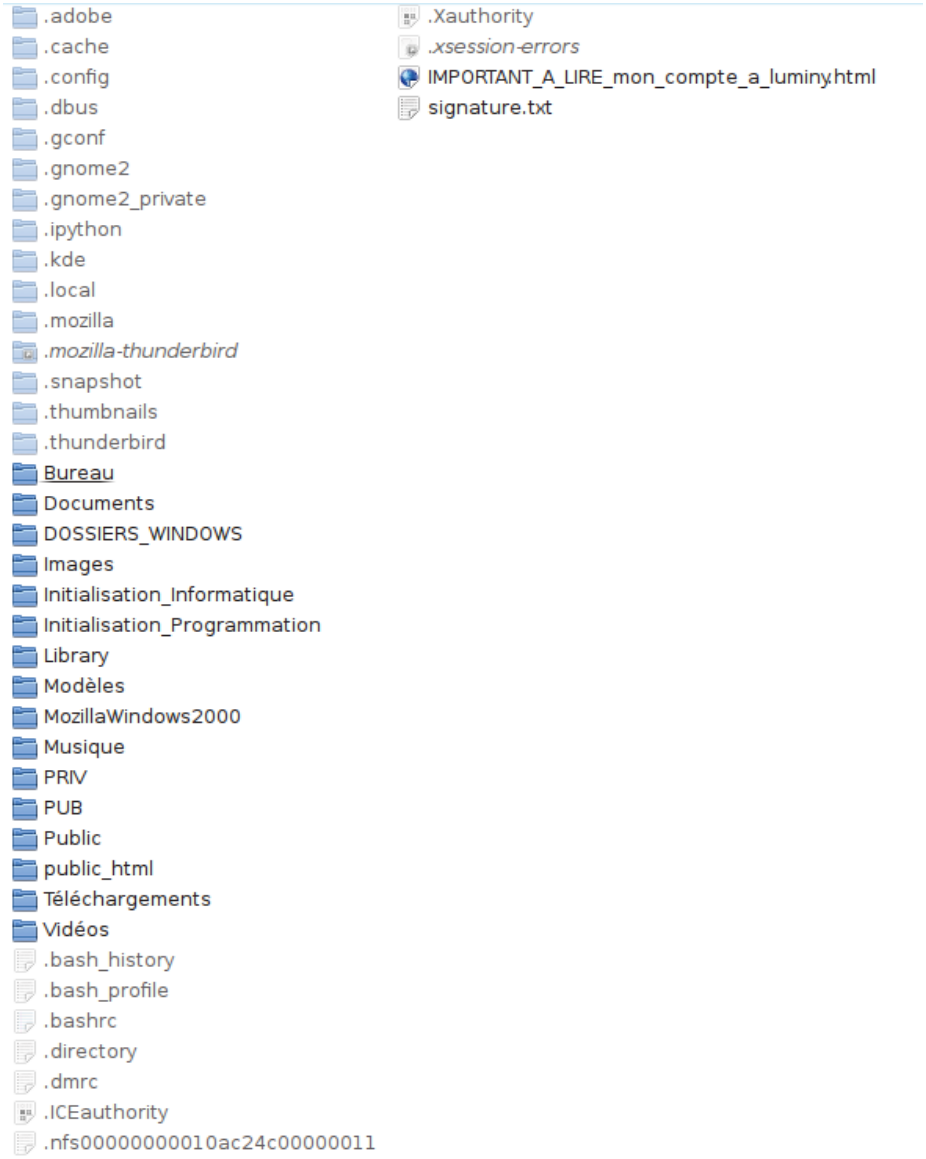
Rapport TP LINUX

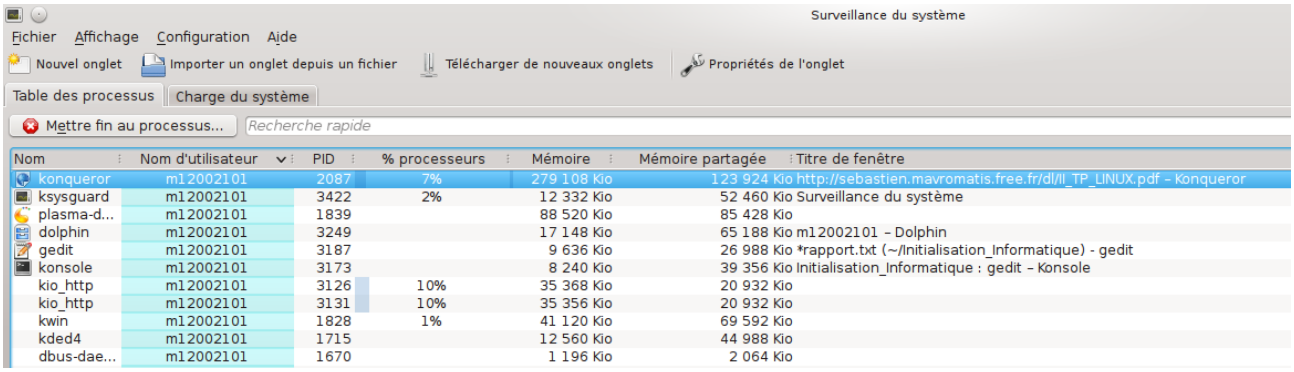
Découverte du système de fichiers Linux via KDE

1. Un clic gauche permet d'ouvrir un fichier/dossier alors qu'un clic droit ouvrir un menu permétant diverses opérations. Un clic central sur un dossier va l'ouvrir dans un nouvel onglet
2. Le chemin de mon dossier $HOME est :

   /filer/etudiants/g9/m12002101
3. Tous les fichiers cachés sous linux commencent par un « . » comme sous l'image ci-dessous (en transparent e sont les fichiers cachés).



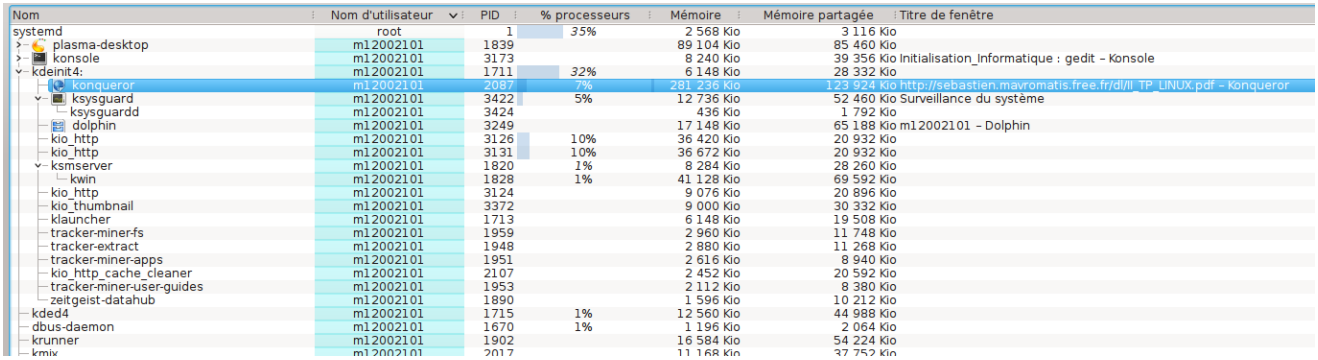
Observation des processus via KDE



2)

- La colonne nom d'utilisateur décrit à quel utilisateur appartient un processus
- La colonne nom décrit à quelle application un processus correspond
- La colonne PID décrit l'identifiant PID d'un processus dans la machine3

3)



5) L'envoi d'un des deux signaux (**SIGTERM** ou **SIGKILL**) provoque la fermeture de l'application dont le PID était visé par le signal. La différence entre ces des signaux c'est que **SIGTERM** demande à application de ce fermer de façon douce, quand à **SIGKILL** provoque un arrêt brutal du programme.

Premier contact avec l'interpréteur de commande

2) La commande *pwd* m'a renvoyé le résultat ci-dessous :

```
m12002101@L-024110A008-03:~$ pwd
/filer/etudiants/g9/m12002101
```

Cette commande renvoie donc le chemin du répertoire courant.

3) La commande *whoami* retourne le nom de l'utilisateur courant. La commande *ps –f –U login* renvoie la liste des processus courant appartenant à l'utilisateur en paramètre.

5) La commande *ls* affiche les fichiers et dossiers du répertoire courant (sauf ceux qui sont cachés).

6)

- *ls -a* : Affiche tous les fichiers et dossiers du répertoire courant (même cacher)
- *ls -al* : Affiche sous forme de liste les fichiers et dossiers du répertoire courant on peut y retrouver notamment les droits sur le fichier, la date de création, la taille.

- *ls -al --color=auto :* Rajoute de la couleur notamment sur les exécutables et dossiers.

7) Pour créer un alias il suffit de taper dans la console : *alias nomdelalias= "nomdelacommande"*

8) Pour ce faire j'ai tapé la commande suivante : alias llc="*ls -al --color=auto*"

10) Cette commande ferme le terminal

11) L'alias n'a pas étais sauvegardé, pour qu'il soit sauvegardé il fallait taper l'alias dans le fichier nommée .bashrc

Création de fichiers

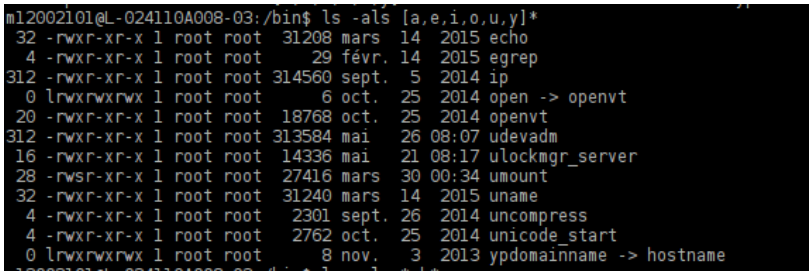
8) En tapant la commande *cat > salut* je me suis rendu compte que lorsqu'on utilise le symbole de redirection > celui-ci écrase le contenu du fichier en commençant au début contrairement à au symbole >> qui écrit le contenu à la fin du fichier.

Notion de filtre de fichiers

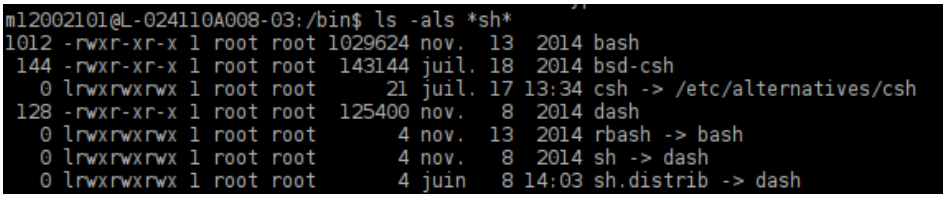
1) *ls *[a,m][a,m]**

3)

- Dont le nom commence par a *: ls a**
- Dont le nom commence par une voyelle : *ls [a,e,i,o,u,y]**

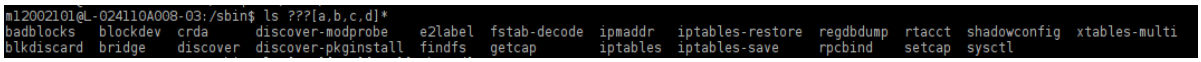


- Dont le nom contient sh : *ls *sh**



- De 3 lettres qui se terminent par sh : *ls ?sh*

4)



Nom de fichiers et caractères spéciaux

1) Le caractère * est considéré comme un joker donc lorsqu'on fait un *ls a*b* il est normal que l'on retrouve les autres fichiers crées préalablement.

2) Si on tape *ls a*b* ou *ls a''*''b* le caractère spécial * n'est plus considéré comme un joker car il est échappé par le \ ou par les doubles apostrophes.

3)

- *mkdir cou\\cou*
- *mkdir cou''\''cou*
- *mkdir ''cou\cou''*

4) *touch mon\ fichier*

Compression/Décompression de fichiers

5) Le fichier memo unix.pdf fait 299884 octets.

Format	ZIP	GZIP
Commande de compression	zip II_MEMO_UNIX.zip II_MEMO_UNIX.pdf	gzip -c II_MEMO_UNIX.pdf > II_MEMO_UNIX.gzip
Commande de décompression	unzip II_MEMO_UNIX.zip	gunzip II_MEMO_UNIX.gzip
Taille de l'archive compressée (en octets)	290056	289909

7)

- Archivage du répertoire TP3 au format TAR : *tar -fc TP3.tar TP3*
- Listage du contenu de l'archive TP3.tar : *tar -tfv TP3.tar*
- Restitution du répertoire TP3 : *tar -fx TP3.tar*

Les droits d'accès

2)

- *ls ./SousRepertoire1* : Permission non accordée
- *ls -l ./SousRepertoire1* : Permission non accordée
- *cd ./SousRepertoire1* : Permission non accordée
- *cat ./SousRepertoire1/fic* : Permission non accordée
- *touch ./SousRepertoire1/fic2* : Permission non accordée
- *rm ./SousRepertoire1/fic* : Permission non accordée
- *ls ./SousRepertoire2* : Permission non accordée
- *cd ./SousRepertoire2* : OK
- *cat ./SousRepertoire2/fic* : OK
- *touch ./SousRepertoire2/fic2 :* Permission non accordée
- *rm ./SousRepertoire2/fic* : Permission non accordée
- *echo ''Comment vas tu'' >> ./SousRepertoire2/fic* : OK

On en déduit que pour faire un *cd* il faut avoir les droits d'exécution, pour faire un *ls* il faut avoir les droits de lecture et d'exécution, pour faire un *touch* il faut le droit d'écriture et d'exécution. Pour faire un *rm* il faut les droits d'écriture et d'exécution, pour faire un *cat* il faut les droits de lecture et d'exécution sur le dossier.

3) Il ne pourra plus rien faire sur son fichier. Il aurait dû mettre les droits suivant pour pouvoir l'utiliser tous en gardant une protection : rwx _ _ _ _ _ _

Gestion des processus

2) Le processus père est le terminal

3) Emacs est arrêté aussi

4) Le processus père a changé c'est un autre processus qui est devenu le père. Le processus qui a adopté emacs est init le processus racine du système.

Redirections

1) *ls -l > toto*

2) ps -A >> toto

3) ps -Af > tata

4)

- *wc -w bonjour*
- *wc -m bonjour*
- *wc -l tata*
- *sort tata*
- *head toto -n10*
- *tail toto –n5*

5) *sort tata | head -n23 | tail -n20 > titi*

Recherche d'une chaine de caractères dans des fichiers textes

1)

- grep -c
- grep -l
- grep -v
- grep -i
- grep -x

2) Le point . correspond à n'importe quel caractère.

Le point d'interrogation ? est utilisé pour indiquer que l'élément précédent est facultatif et peut être rencontré au plus une fois.

L'étoile * est utilisée pour indiquer que l'élément précédent peut être rencontré zéro ou plusieurs fois.

3)

- *grep ''^.o'' tata* : Prend tous les lignes qui commence par un caractère puis un suivit d'un o
- grep -ni bash tata : Affiche les lignes contenant bash en ignorant la casse en affichant la ligne
- grep -ni *''td$'' tata* : Affiche les lignes finissant par td en ignorant la casse en affichant la ligne

4) *grep ''^r.*[0-9]$''*

5) ps -Af | grep *''^m12002101.*''*

6) ps -Af | grep *''^m12002101.*' | wc -l'*

Recherche de fichiers dans une arborescence

1) La commande *locate* utilise une base de donne crée par la commande *updatedb*, alors que *find* fait une recherche brut sur le disque.

2) *ls -R -l | grep ''nomfichier$''*

Les variables d'environnement

1)

- $HOME : Chemin du répertoire de l'utilisateur courant
- $USER : Nom de l'utilisateur courant
- $TERM : Affiche le nom de l'émulateur de terminal courant
- $HOSTNAME : L'adresse de la machine courante (a vérifier)
- $SHELL : Le chemin du terminal courant
- $PATH : Affiche les différents chemins des commandes notamment

2)

- touch fichier $USER : Créer deux fichier un nommé fichier et l'autre avec le contenu de la variable $USER
- touch fichier $USER : Crée un fichier nommé ''fichier m12002101'' dans mon cas.
- ''touch fichier $USER'' : Ce n'est pas une commande donc elle ne fera rien à part une erreur.

Donc les ‘ et '' permettent d'échapper les caractères spéciaux ou commande tous en gardant le contenu des variables d'environnement.

3)

- TEMP2=*''ma premiere variable''*
- TEMP3=$PATH
- $PATH et $TEMP3 sont bien égaux
- $PATH=*''salut''*
- La commande *ls* n'a pas été trouvé car le terminal ne sait plus où se trouve la commande sur le disque.
- PATH=$TEMP3
- La variable d'environnement $PATH permet d'indiquer au terminal le chemin des différentes commandes accessible. Sans elle pour il faudra taper le chemin complet de l'exécutable de la commande.

Les arguments des scripts

1)

```
#!/bin/bash
echo $0 $1 $2 $3 $4
```

2)

```
#!/bin/bash
echo ls $1
```

3)

```
#!/bin/bash
mkdir $1
touch $2
chmod 222 $1/$2
```

Les conditionnelles dans les scripts

2)

```
#!/bin/bash
if [ $1 -e ]
then
        cat $1
else
        echo ''Le fichier n'existe pas''
fi
```

Les boucles dans les scripts

2)

```
#!/bin/bash
for fichier in ./*
do
        if [ -d $fichier ]
        then
                echo ''$fichier est un répertoire''
        else
                echo ''$fichier n'est pas un répertoire''
        fi
done
```

3)

```
#!/bin/bash
for fichier in ./*
do
        if [ -d $fichier ]
        then
                echo ''$fichier :''
                cd $fichier
                echo `ls`
                cd ..
        else
                echo $fichier
        fi
done
```