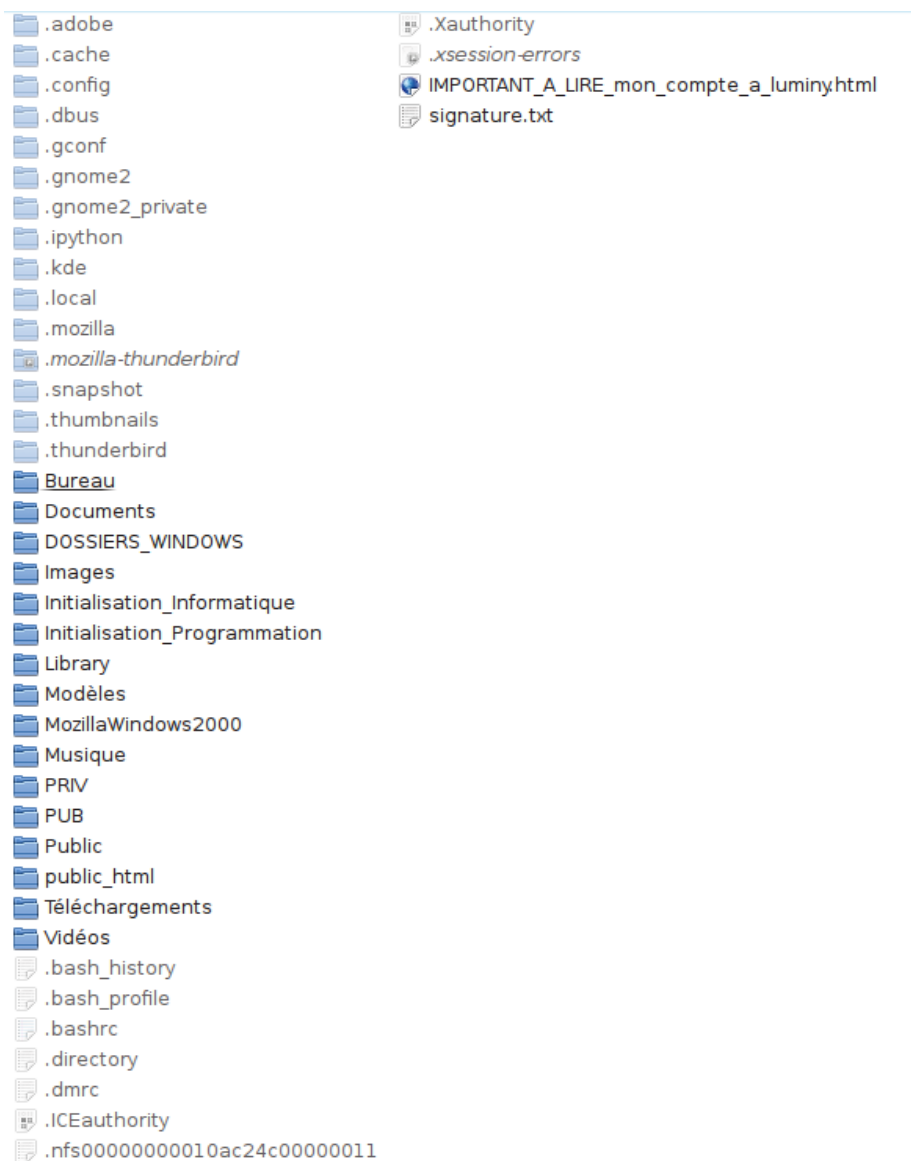


Rapport TP LINUX

Découverte du système de fichiers Linux via KDE

1. Un clic gauche permet d'ouvrir un fichier/dossier alors qu'un clic droit ouvre un menu permettant 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

Surveillance du système

Fichier Affichage Configuration Aide

Nouvel onglet Importer un onglet depuis un fichier Télécharger de nouveaux onglets Propriétés de l'onglet

Table des processus Charge du système

Mettre fin au processus... Recherche rapide

Nom	Nom d'utilisateur	PID	% processeurs	Mémoire	Mémoire partagée	Titre de fenêtre
konqueror	m12002101	2087	7%	279 108 Kio	123 924 Kio	http://sebastien.mavromatis.free.fr/dl/TP_LINUX.pdf - Konqueror
ksysguard	m12002101	3422	2%	12 332 Kio	52 460 Kio	Surveillance du système
plasma-d...	m12002101	1839		88 520 Kio	85 428 Kio	
dolphin	m12002101	3249		17 148 Kio	65 188 Kio	m12002101 - Dolphin
gedit	m12002101	3187		9 636 Kio	26 988 Kio	*rapport.txt (~/.Initialisation_Informatique) - gedit
konsole	m12002101	3173		8 240 Kio	39 356 Kio	Initialisation_Informatique : gedit - Konsole
kio_http	m12002101	3126	10%	35 368 Kio	20 932 Kio	
kio_http	m12002101	3131	10%	35 356 Kio	20 932 Kio	
kwin	m12002101	1828	1%	41 120 Kio	69 592 Kio	
kdcd4	m12002101	1715		12 560 Kio	44 988 Kio	
dbus-dae...	m12002101	1670		1 196 Kio	2 064 Kio	

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 machine

3)

Nom	Nom d'utilisateur	PID	% processeurs	Mémoire	Mémoire partagée	Titre de fenêtre
systemd	root	1	35%	2 568 Kio	3 116 Kio	
plasma-desktop	m12002101	1839		89 104 Kio	85 460 Kio	
konsole	m12002101	3173		8 240 Kio	39 356 Kio	Initialisation_Informatique : gedit - Konsole
kdeinit4	m12002101	1711	32%	6 148 Kio	28 332 Kio	
konqueror	m12002101	2087	7%	281 236 Kio	123 924 Kio	http://sebastien.mavromatis.free.fr/dl/TP_LINUX.pdf - Konqueror
ksysguard	m12002101	3422	5%	12 736 Kio	52 460 Kio	Surveillance du système
ksysguardd	m12002101	3424		436 Kio	1 792 Kio	
dolphin	m12002101	3249		17 148 Kio	65 188 Kio	m12002101 - Dolphin
kio_http	m12002101	3126	10%	36 420 Kio	20 932 Kio	
kio_http	m12002101	3131	10%	36 672 Kio	20 932 Kio	
kmsserver	m12002101	1820	1%	8 284 Kio	28 260 Kio	
kwin	m12002101	1828	1%	41 128 Kio	69 592 Kio	
kio_http	m12002101	3124		9 076 Kio	20 896 Kio	
kio_thumbnail	m12002101	3372		9 000 Kio	30 332 Kio	
klauncher	m12002101	1713		6 148 Kio	19 508 Kio	
tracker-miner-fs	m12002101	1959		2 960 Kio	11 748 Kio	
tracker-extract	m12002101	1948		2 880 Kio	11 268 Kio	
tracker-miner-apps	m12002101	1951		2 616 Kio	8 940 Kio	
kio_http_cache_cleaner	m12002101	2107		2 452 Kio	20 592 Kio	
tracker-miner-user-guides	m12002101	1953		2 112 Kio	8 380 Kio	
zeitgeist-datahub	m12002101	1890		1 596 Kio	10 212 Kio	
kdcd4	m12002101	1715	1%	12 560 Kio	44 988 Kio	
dbus-daemon	m12002101	1670	1%	1 196 Kio	2 064 Kio	
krunner	m12002101	1902		16 584 Kio	54 224 Kio	
kmix	m12002101	2017		11 168 Kio	27 752 Kio	

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 deux signaux c'est que **SIGTERM** demande à l'application de se 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 cachés)
- `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 été sauvegardé, pour qu'il soit sauvegardé il fallait taper l'alias dans le fichier nommé `.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]*`

```
m12002101@L-024110A008-03:/bin$ ls -als [a,e,i,o,u,y]*
32 -rwxr-xr-x 1 root root 31208 mars 14 2015 echo
4 -rwxr-xr-x 1 root root 29 févr. 14 2015 egrep
312 -rwxr-xr-x 1 root root 314560 sept. 5 2014 ip
0 lrwxrwxrwx 1 root root 6 oct. 25 2014 open -> openvt
20 -rwxr-xr-x 1 root root 18768 oct. 25 2014 openvt
312 -rwxr-xr-x 1 root root 313584 mai 26 08:07 udevadm
16 -rwxr-xr-x 1 root root 14336 mai 21 08:17 unlockmgr_server
28 -rwsr-xr-x 1 root root 27416 mars 30 00:34 umount
32 -rwxr-xr-x 1 root root 31240 mars 14 2015 uname
4 -rwxr-xr-x 1 root root 2301 sept. 26 2014 uncompress
4 -rwxr-xr-x 1 root root 2762 oct. 25 2014 unicode_start
0 lrwxrwxrwx 1 root root 8 nov. 3 2013 ydomainname -> hostname
```

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

```
m12002101@L-024110A008-03:/bin$ ls -als *sh*
1012 -rwxr-xr-x 1 root root 1029624 nov. 13 2014 bash
144 -rwxr-xr-x 1 root root 143144 juil. 18 2014 bash-csh
0 lrwxrwxrwx 1 root root 21 juil. 17 13:34 csh -> /etc/alternatives/csh
128 -rwxr-xr-x 1 root root 125400 nov. 8 2014 dash
0 lrwxrwxrwx 1 root root 4 nov. 13 2014 rbash -> bash
0 lrwxrwxrwx 1 root root 4 nov. 8 2014 sh -> dash
0 lrwxrwxrwx 1 root root 4 juin 8 14:03 sh.distrib -> dash
```

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

4)

```
m12002101@L-024110A008-03:/sbin$ ls ???[a,b,c,d]*
badblocks blockdev crda discover-modprobe e2label fstab-decode ipmaddr iptables-restore regdbdump rtacct shadowconfig xtables-multi
blkdiscard bridge discover discover-pkginstall findfs getcap iptables iptables-save rpcbind setcap sysctl
```

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 créés 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	<code>zip II_MEMO_UNIX.zip II_MEMO_UNIX.pdf</code>	<code>gzip -c II_MEMO_UNIX.pdf > II_MEMO_UNIX.gzip</code>
Commande de décompression	<code>unzip II_MEMO_UNIX.zip</code>	<code>gunzip II_MEMO_UNIX.gzip</code>
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 chaîne 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 données créée 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 (à vérifier)
- \$SHELL : Le chemin du terminal courant
- \$PATH : Affiche les différents chemins des commandes notamment

2)

- touch fichier \$USER : Créer deux fichiers 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 tout 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 trouvée 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 accessibles. 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
```