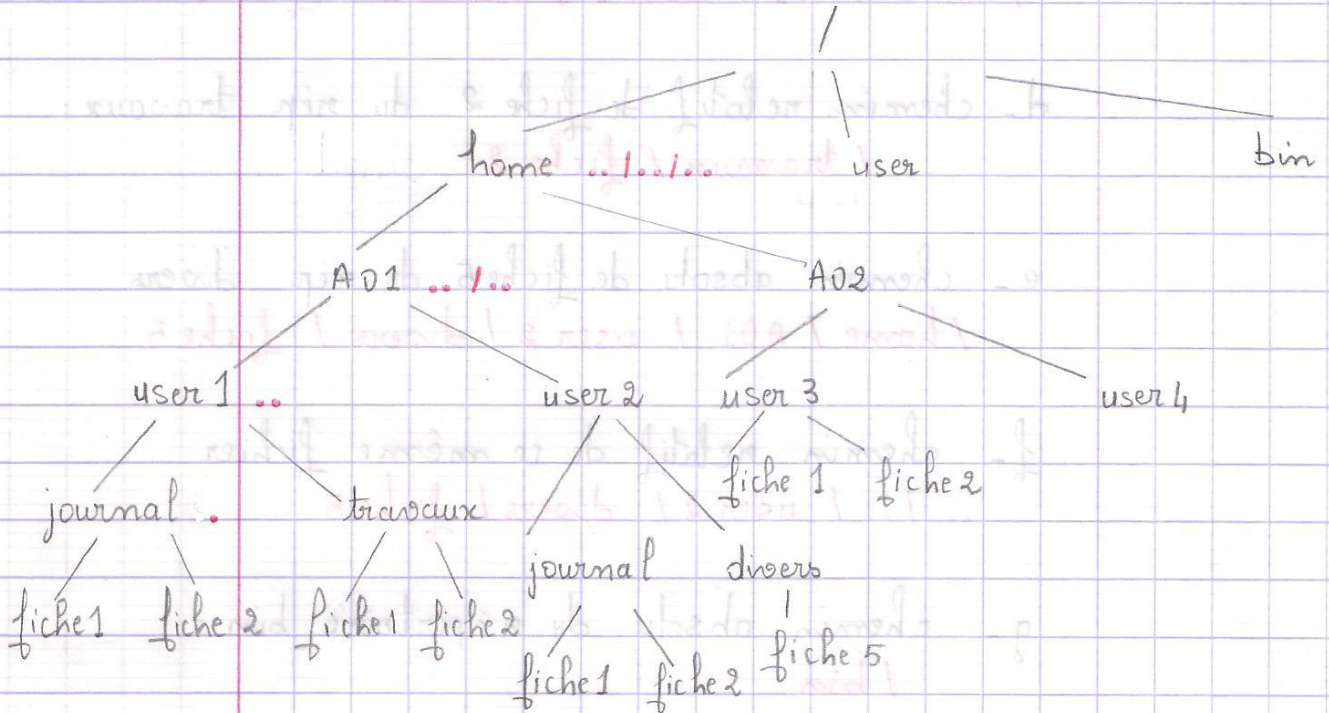


Système d'exploitation

18 - 09 - 2014

TD



chemin absolue

=> c'est toujours par rapport à la racine

chemin relatif

=> par rapport au répertoire courant

.. répertoire parent

. répertoire courant

L'utilisateur se trouve actuellement dans le sous-répertoire journal du répertoire user 1

I / - Donner les chemins suivants :

a - chemin absolu du répertoire courant :

/ home / A01 / user 1 / journal

b - chemin relatif de user 1 :

..

c. chemin absolu de fiche 1 du rép travaux :
/home / A01 / user 1 / travaux / fiche 1

d. chemin relatif de fiche 2 du rép travaux :
.. / travaux / fiche 2

e. chemin absolu de fiche 5 du rép divers
/home / A01 / user 2 / divers / fiche 5

f. chemin relatif de ce même fichier
.. / .. / user 2 / divers / fiche 5

g. chemin absolu du répertoire bin :
/bin

h. chemin relatif de ce même répertoire
.. / .. / .. / bin

i. chemin absolue de user 3
/home / A02 / user 3 /

j. chemin relatif de user 3
.. / .. / .. / A02 / user 3

II / - Ecrire les commandes permettant les actions suivantes (sans changer le rép. de travail)

a. lister le contenu du répertoire courant:

ls (ou bien ls.)

b. lister le contenu du rép. travaux:

ls .. / travaux

c. afficher le chemin absolu du rép. courant

pwd

d. créer un répertoire lundi:

mieux
mkdir lundi (ou bien mkdir .. / lundi)

attention: mkdir / lundi: le rép. va être créé dans la racine

e. créer un fichier vide cr.txt

touch cr.txt

f. copier fiche 1 dans travaux en le nommant fiche 3

cp fiche 1 .. / travaux / fiche 3

source

destination

g. copier fiche 5 dans le répertoire courant:

cp .. / .. / user 2 / divers / fiche 5 .

source

Ps: le chemin peut être absolu ou relatif

h/ renommer ce fichier en fichier_user2

(l'import de user) `mv fiche5 fiche_user2`

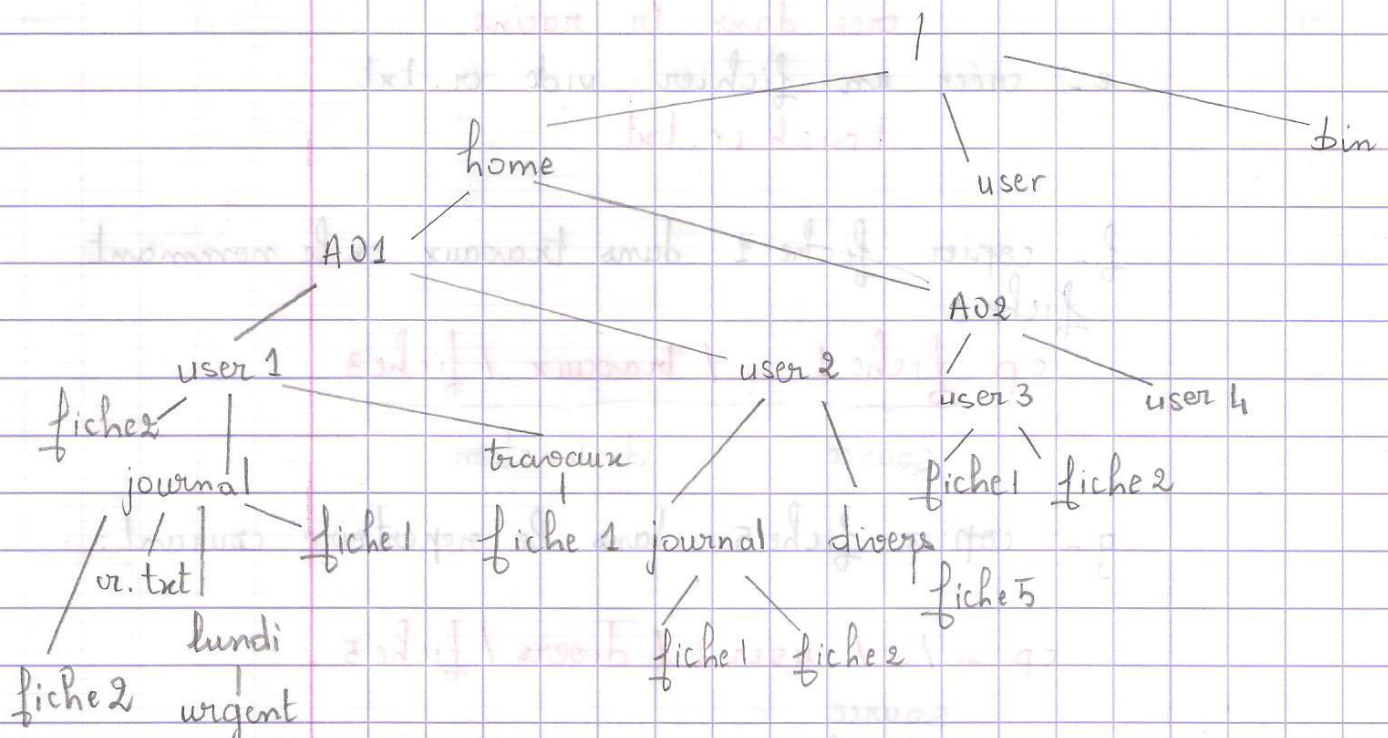
i/ déplacer fiche2 du répertoire courant dans le répertoire user1

`mv fiche2`

j/ déplacer fiche2 du répertoire travaux dans le répertoire lundi en le renommant urgent

`mv ../travaux/fiche2 lundi/urgent`

La nouvelle arborescence :



2-10-2014

exercice 3 : utilisation des droits d'accès

- riri et fifi appartiennent au même groupe (1)
 - loulou ∈ groupe 2 (autres)
 - " /tmp " contient " correction_projet " accessible en lecture par tous les utilisateurs
 - Le répertoire " partage " contient un fichier " projet "
 - Le répertoire de travail est le rép " partage "
- répertoire courant

configuration 1:

a) cp /tmp/correction_projet projet

droit d'accès de lecture

droit d'accès de
"modification" écriture

niri - fifi - loulou

b) cp / tmp / correction_projet projetbis

on doit avoir les droits
d'accès d'écriture sur
"partage"

configuration 1 (2 & 3)

```
cd chmod 666 projet
```

↑

(nini)

seul le propriétaire qui
peut exécuter la
commande "chmod"

~~fifi~~ - ~~loulou~~

d/ rm projet

niri - ~~fifi~~ - loulou

configuration 2

a/ cp / tmp / correction_projet projet

niri - ~~fifi~~ - loulou

b/ cp / tmp / correction_projet projetbis

niri - ~~fifi~~ - loulou

c/

d/ rm projet

niri - ~~fifi~~ - loulou

configuration 3

a/ cp / tmp / correction_projet projet
→ cucun

b/ cp / tmp / correction_projet projetbis

niri - ~~fifi~~ - loulou

dl rm projet
riri - fifi - loulou

droit d'exécution sur les répertoires seulement
si on veut modifier de repertoire de travail.

drw - rw - rw - partage

\$ cd partage

affiche un message d'erreur :

"vous n'avez pas de droit d'exécution sur le
répertoire partage."

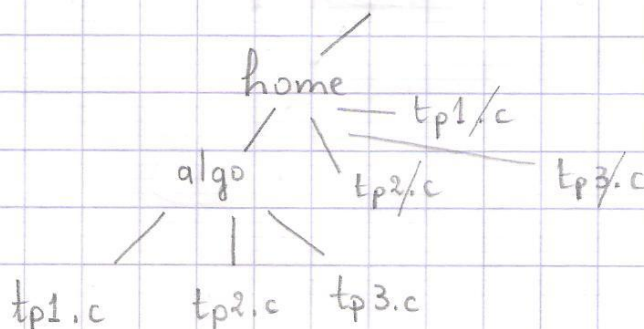
exercice supplémentaire :

1. Que font les commandes suivantes, si "algo"
est un sous repertoire du repertoire courant :

1° mv tp1.c algo

2° mv tp2.c algo

3° mv tp3.c algo



2. si "algo" est un fichier texte du répertoire courant.

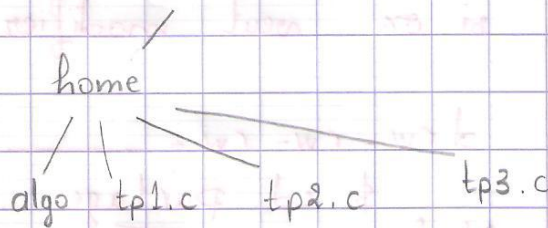
Quel est l'effet de ces trois mêmes commandes?

1°/ mv tp1.c algo renommer tp1.c en algo

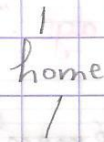
2°/ mv tp2.c algo

3°/ mv tp3.c algo

avant



après



algo ← qui contient le "contenu" de tp3.c

16 - 10 - 2011

exercice 2 (La commande chmod)

rw-r--r-- toto
1/ nouveaux droits d'accès
rwx r-x --- toto
texte symbolique

	utilisateurs	actions	droit
user	u (propriétaire)	+ (ajout d'un droit)	r (lecture)
group	g (groupe de prop)	- (suppression ...)	w (écriture)
other	o (autre)	= (initialisation ...)	x (exécuter)
all	a (tout le monde)		

exemple :

chmod go-x fichier

=> suppression du droit d'exécution pour un groupe ou pour autre

chmod u+r-w fichier

=> ajout du droit de lecture et

suppression du droit d'écriture

chmod a-x fichier

=> suppression du droit d'exécution pour tout le monde
a

u	g	o
$u \ w \ \boxed{-}$	$1 \ - \ \boxed{-}$	$u \ - \ -$
$r \ w \ \boxed{x}$	$u \ - \ \boxed{x}$	$\boxed{-} \ - \ -$
$+ \ x$	$+ \ x$	$- \ x$

toto

\$ chmod ug + x toto

\$ chmod o - x toto

2) chmod mod1, mod2... toto

a) \$ chmod ug + x, o - x toto

b) \$ chmod u = rwx, g = rx, o = toto

3) écriture numérique (octale)

	u	g	o
symbole	rwx	$u - x$	$---$
binnaire	111	101	000
octal	7	5	0

$$u \rightarrow 4 = 2^2$$

$$w \rightarrow 2 = 2^1$$

$$x \rightarrow 1 = 2^0$$

\$ chmod 750 toto

exercice supplémentaire

1. Liens physiques :

Lorsque vous modifiez les données dans un fichier qui possède des liens physiques :

a) juste les données de ce fichier sont affectées,
b) juste les données de ce fichier et les données de ses liens physiques du même répertoire sont affectées.

c) Les données de ce fichier et les données de ses liens physiques sont affectées puisqu'ils ont des i-nœuds différents

d) Les données de ce fichier et les données de liens physiques puisqu'ils partagent le même i-nœud

Rq: Lien symbolique aura ses propres i-nœuds

2) Les commandes `chgrp`, `chown` :

justification :

concernant

a et b

⇒ ce n'est plus le propriétaire du fichier

un utilisateur utilise la commande `chgrp` pour donner la propriété d'un fichier à un deuxième utilisateur

Que doit faire l'utilisateur initial pour reprendre la propriété du fichier ?

a) l'utilisateur initial exécute à nouveau la commande "`chgrp`" en précisant que c'est lui le nouv propriétaire

b) le nouveau propriétaire exécute la commande "`chgrp`" et précise que l'utilisateur initial est le nouveau propriétaire

c) L'utilisateur initial exécute la commande "`chown`" en précisant que c'est lui le nouveau propriétaire

3. La commande "umask"

après l'exécution de la commande `umask 731`

Quels seront les droits d'accès sur les fichiers manuellement créés?

a/ `rw-rw-rw--`

b/ `rw-rw-r--`

c/ `---r--rw-`

d/ `---wx--x`

Rq: `ls -ld Rep`

↳ affiche les informations sur le répertoire "Rep"

4. La commande "chmod"

un fichier "fichier1" a les droits suivants:

`r-----x-w`

la commande `chmod 143 fichier1` aura le même effet que:

a. `chmod u+r-x, g+r-x, o+w fichier1`

b. `chmod u-w, g=rw, o=rx fichier1`

c. `chmod u-r-w, g+r-w, o+r-x fichier1`

d. `chmod a=x, g=r, o=wx fichier1`

e. `chmod u+w, g+r-w, o+r-x fichier1`

f. `chmod u=rw, g=r, o=r fichier1`

30-10-2014

exercice 4 : (Programmation Shell)

Question (1, 2, 3, 5)

\$ cmd (Le système cherche cette commande dans les répertoires définis dans la variable d'environnement PATH)

\$ ls-l aucune commande avec le nom ls-l

(PATH doit contenir le chemin vers le répertoire courant)

1) ./usr/bin

- export /home/
- /bin

→ PATH = \$PATH : ./usr/bin : /export/home/ : /bin

on utilise les deux points
pour séparer les chemins

attention!

PATH = /user/bin : /export/home/ : /bin

⇒ cette commande va écraser le contenu de la variable PATH.

variable = valeur ←
pour accéder à la valeur
d'une variable on utilise \$
echo \$variable

pour affecter on
utilise pas \$ sauf
si on veut affecter
le contenu d'une
autre variable

ex : x = 123

variable = \$x

echo \$variable ⇒ 123

\$ which ls) le chemin vers le
=> /bin/ls) répertoire qui contient
cette commande

21

utilité de & : exécution d'une commande
en arrière plan.

comment remettre cette commande en premier plan ?

=> \$fg

exemple 1:

\$ cmd & ← execution de cmd en arr plan

\$fg ← remet cmd en prem plan

exemple 2:

cmd1 & ; cmd2 & ; cmd3 &

[1] ——— cmd 1

[2] ——— cmd 2

[3] ——— cmd 3

↑

numéro de la tâche

\$fg 2 ← numéro de la tâche à mettre en
premier plan

ctrl + Z
\$bg

} pour la remettre en arrière plan.

• cmd1 & ; cmd2 & ; cmd3

⇒ cmd1 et cmd2 sont exécutées en arrière plan et cmd3 est exécuté en premier plan.

• cmd1 & ; cmd2 & ; cmd3 &

⇒ les trois commandes sont exécutées en arrière plan

3. Quelle est la diff entre:

• cmd 1 1 > sortie.txt 2 > & 1

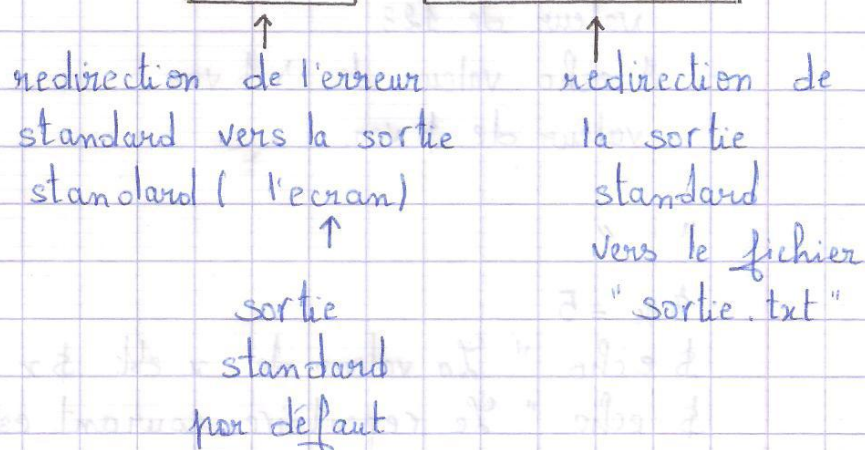
• cmd 1 2 > & 1 1 > sortie.txt

• cmd 1 1 > sortie.txt 2 > & 1



conclusion : la sortie standard et l'erreur standard sont redirigées vers le fichier "sortie.txt"

• cmd 1 2 > & 1 1 > sortie.txt



exemple:

```
$ ls -l archive 2>&1 1> sortie.txt
```

↑

fichier

n'existe

nois

→ affichage erreur sur l'écran

```
$ cat sortie.txt
```

(rien)

```
$ls -l 2>&1 1>sortie.txt
```

\$ cat sortie.txt

$$\begin{pmatrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{pmatrix}$$

"listing" du contenu
du répertoire courant

5.

1: neutraliser la signification spécial du caractère (\$, * , ? , & , !) qui le suit

example :

\$ var = 123

\$ echo valeur de \$ var

valeur de 123

\$ echo valeur de 1 \$ var

valeur de $\$$ var

$$\$x = 5$$

\$ echo "La valeur de x est \$x"

```
$ echo "Le repertoire courant est 'pwd'"
```

la valeur de x est 5

le répertoire courant est /home/compte1

$\Rightarrow$ ça neutralise la signification des caractères
et elle permet la substitution des variables
et des commandes.

' ' $\Rightarrow$ ça neutralise la signification mais
ne substitue pas les var et les commandes

ex: \$ echo 'la valeur de x est \$ x '

$\Rightarrow$ La valeur de x est \$ x

exercice supplémentaire: (Script Shell)

écrire un script Shell (NomPrenom.sh)
qui demande à l'utilisateur la saisie de son
nom et de son prénom et ensuite affiche:
Bonjour Nom Prénom
votre répertoire courant est : /home/comptel
ce répertoire contient les fichiers suivants:

_____) contenu du
_____) répertoire courant

Réponse:

```
# ! /bin/bash (ou bien /bin/sh)
```

le chemin vers l'interpréteur (pear - awk...)

```
echo "Donner votre nom & prénom"
```

= il est neutraliser

```
read $nom $prenom read nom prenom
```

```
echo "Bonjour $nom $prenom"
```

↳ le nom de la

```
echo "votre répertoire courant est: 'pwd'"
```

var pas son

```
echo "Ce répertoire contient les \ "
```

contenue!

```
fichiers suivants: 'ls -l' "
```

neutraliser entier
↑

NomPrenom.sh

```
$ ./NomPrenom.sh erreur (il faut le rendre executable) execution
```

```
chmod u+x NomPrenom.sh
```

```
$ ./NomPrenom.sh ou $ /home/comptel/NomPrenom.sh
```

les droits nécessaires pour l'exécution d'un script sont la lecture
et l'exécution

27 - 11 - 2014

Syntaxe du "if"

if [expression 1] ($\Leftrightarrow$ test expression 1)

then

Liste de commandes 1 (si expression 1 est vrai (1))

elif [expression 2]

then

Liste de commandes 2 (si expression 1 est fausse
et expression 2 est vrai)

elif [expression n]

then

Liste de commandes n (si $\forall i=1, \dots, n-1$ expression i
est fausse et expression n est vrai)

else

Liste de commandes n+1

(si $\forall i=1, \dots, n$ expression i fausse et expression n+1
est vrai)

fi

notes :

expression peut être une commande !

-e \$fich (existence de l'objet \$fich)

-d \$fich (\$fich existe et est un répertoire)

suite de l'exercice 4:

6.

```
# ! /bin / bash
echo -e "entrez le nom de rep 1c"
read REP
if [ ! -d $REP ] ; then
    mkdir $REP
fi
cd $REP
echo "On est dans le repertoire"
```

si j'écris le if et le then sur la m^{ême} ligne faut les séparer par un point-virgule

```
echo "entrez le nom du rep"
read rep
```

écran
entrez le nom du rep
\$ Image (clavier)

```
read -p "Entrez le nom du repertoire" rep
```

```
echo -e "entrez le nom du rep 1c"
read REP
```

écran
entrez le nom du rep Image
clavier

Remarque :

```
$ REP = Images
$ mkdir REP => crée un rép avec le nom REP
$ mkdir $REP => crée le répertoire Images
```

↳ contenu de la variable

7.

```
# ! / bin / bash
echo -e "entrez un nom pour votre fich ic"
read fich
if [-e $fich]
then echo "$fich existe"
      chmod go-rwx $fich
else
      echo "$fich n'existe pas"
fi
```

lecture
à partir
du
clavier

./ protege Images

paramètres de position

\$1

argument au script

script partage |

```
# ! / bin / bash
if [-e $1]
then
  chmod og-rwx $1
fi
```

la racine de votre compte

(compte de connexion) ~

(répertoire personnel)

~/poubelle

\$ cd \$HOME $\Leftrightarrow$ \$ cd ~

un chemin
relatif

$\Leftrightarrow$ \$ cd /home / compte 1

$\Leftrightarrow$ \$ cd

8.

```
# ! /bin / bash
read -p "entrez le nom du fich " fich
if [ -f $fich ] ; then
    if [ ! -d ~/poubelle ] ; then
        mkdir ~/poubelle
    fi
    mv $fich ~/poubelle
fi
```

le 11 - 10 - 2014

Système d'exploitation TP1

installation et utilisation du système Unix

identifiant : compte 1
mot de passe : compte1

renommer prof → prof 2
SE → SE2

2. ls -l

3. mkdir prof2
ls -l prof2

4. mkdir SE2

5. cd SE2

6. touch smi_al

7. cat > smi_d

smi = sci. mathématiques informatique, année
scolaire 2014 - 2015

ctal D

8. cat smi

9. mkdir TP1

10. ls -l

11. ls -al

12. rm smi_al

13. mv smi toto

14. cp toto /home/compte 1 / prof2

15. ls prof2

16. mv toto TP1

17. rmdir TP1

→ le répertoire n'est pas vide !

18. man tar (pour + d'info sur la com tar)

La rép → tar -cf /home / compte 1 / prof 2 / archive tar SE2/ TP

tar -tvf /home / compte 1 / prof 2 / archive tar