Volatiana RALAMBONDRAINY – Samy FOUILLEUX – Maxime CHAMBREUIL
ASI 3 – Année 2001 / 2002

Architecture des Ordinateurs & Systèmes d'Exploitation
TP 9

Objectifs :

- Comparer le système de fichiers de 2 SE (Windows 98 et Linux).
- Manipuler des commandes de gestion de fichiers sous Unix.
- Création de scripts Bash.

I) Comparaison de systèmes de fichiers de Windows 98 et Linux

	Windows 98	Linux
Noms de fichiers :	Longueur limitée à 180 caractères parmi 228. Pas de différence entre minuscules et majuscules : Ex.exe et ex.exe ne peuvent pas exister dans le meme repertoire. Les extensions permettent de savoir quel logiciel utiliser pour ouvrir ou editer le fichier.	Longueur limitée à 230 caractères parmi 255 Noms différents avec ou sans majuscules Il n'y pas d'extension sous Unix. Tout est fichier.
Types de fichiers possibles :	Ordinaire Dossier (Répertoire)	Ordinaire Répertoire Périphériques accédé en mode caractère, bloc ou tube Lien symbolique Socket
Attributs existants :	Nom du fichier Type Emplacement Taille Date de création, modification et dernière accession Lecture seule / caché / système / archive	Nom du fichier Permissions d'accès Propriétaire, groupe Date de modification et dernière accession Taille
Arborescence de catalogues :	Selon les lecteurs Affichage par nom, type, taille ou date de création	A partir de la racine Affichage par nom
Lecture de système de fichiers :	Win 98 ne peut pas lire d'autres systèmes de fichiers (Linux).	Linux peut lire la partition Win98 (FAT32) : il crée un point de montage.
Allocation des fichiers :	Win98 empile les fichiers sur le disque. Un fichier supprimé = un trou donc défragmentation nécessaire.	Système i-node : défragmentation des fichiers inutile

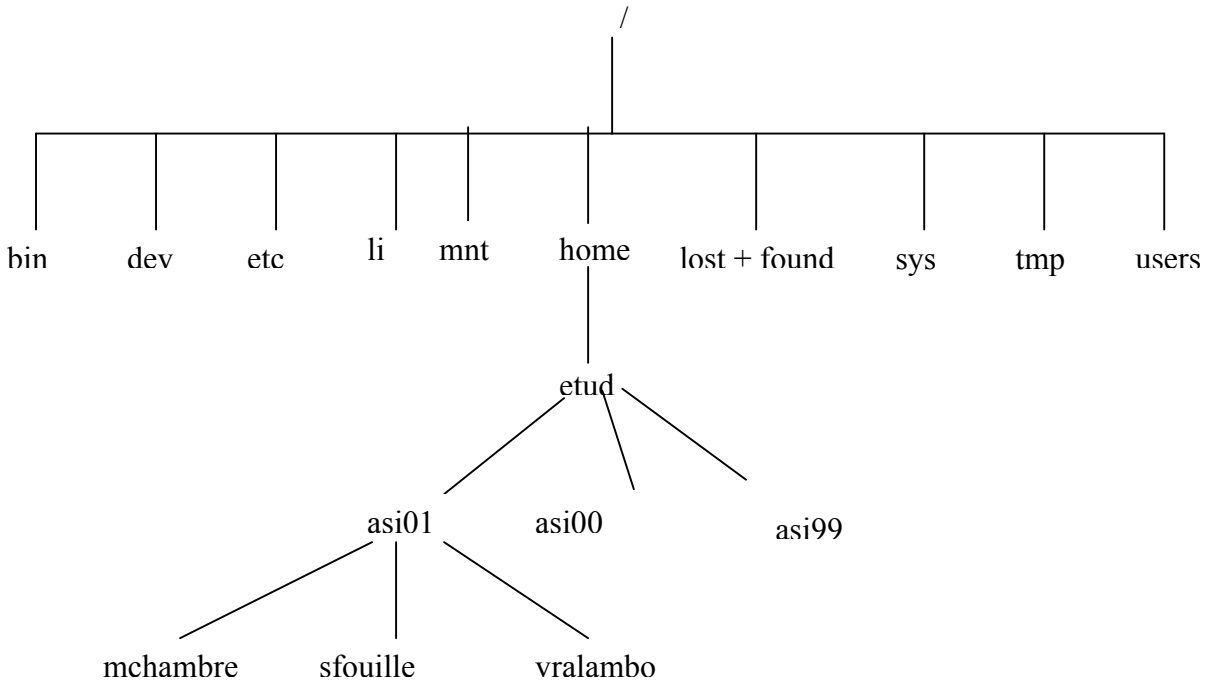
	Utilitaire de défragmentation puissant fourni.	
Gestion des fichiers partagés :	Win 98 gère ces fichiers sous forme de lien symbolique	Fichier partagé consultable et mise à jour possible.
Fiabilité (réparer disque, gérer archives) :	Réparation de disque : Scandisk Archivage : Winzip et lecteur Zip	Réparation de disque : fsck Archivage : tar
Sécurité :	Tous les fichiers sont accessibles à n'importe quel utilisateur (même les plus sensibles du SE). L'utilisation de logins différents ne sert qu'à personnaliser le papier peint du bureau, les raccourcis du menu démarrer, le dossier Mes Documents, etc...	Les permissions permettent de définir les utilisateurs pouvant accéder au fichier.

Exercice 2 : Systèmes de fichiers UNIX

1 – Exploration

a - Les types de fichiers

Arborescence du système de fichiers UNIX



- Fichier répertoire : vralambo

exemple :
drwxr-xr-x 30 vralambo asi01 4096 nov 27 11:27 vralambo

- Fichier ordinaire :

exemple :
-rw-r--r-- 1 vralambo asi01 12963 nov 8 23:06 exo2.rtf
-rw-r--r-- 1 vralambo asi01 708 nov 8 22:01 sscha.asm

- fichiers l lien symbolique :

C'est un entrée d'un répertoire interprété directement comme une chaîne de caractères.
C'est un nom de chemin relatif.
exemples de fichiers :
lrwxrwxrwx 1 root root 19 fév 12 2001 libthread_db.so.1 -> lib
lrwxrwxrwx 1 root root 19 fév 12 2001 libthread_db.so.1 -> lib

Il existe plusieurs types de fichiers spéciaux :

- les fichiers b en mode bloc (b) :

Ils servent à modéliser les périphériques tels que les disques constitués d'un certain nombre de blocs adressables individuellement. (on en trouve dans /dev par ex.)

exemples de fichiers :
crw------- 1 root sys 68, 3 aoû 24 2000 capi20.02
crw------- 1 root sys 68, 4 aoû 24 2000 capi20.03

- les fichiers c en mode caractère :

Ils servent à modéliser les périphériques qui gèrent les flots de caractères plutôt que les blocs directement adressables. (exemple : imprimantes, terminaux, …)
exemples de fichiers :
crw------- 1 root sys 68, 3 aoû 24 2000 capi20.02
crw------- 1 root sys 68, 4 aoû 24 2000 capi20.03

- fichiers p tube (tube de communication) :

C'est un pseudo-fichier qui peut être utilisé pour relier 2 processus.
exemples de fichiers :
prw------- 1 root root 0 oct 4 12:33 initctl

- fichiers s socket :

Ces fichiers aident à la communication entre terminaux et agissent comme des boîtes aux lettres. Ces fichers font la « connexion » entre un émetteur et un récepteur. On en trouve dans /dev
exemples de fichiers :
srwx------ 1 root root 0 oct 4 12:16 gpmctl

Remarque : tty indique le nom de la fenêtre et de l'ordinateur ou l'on travaille. Pour UNIX, chaque fenêtre est un fichier. Ces fenêtres sont dans le répertoire /dev
Pour afficher le résultat d'une commande tapée dans le 1ère fenêtre et l'afficher dans le 2ème on tape : ls -al > /dev/pts/2

b - Les droits d'accès

UNIX possède des mécanismes permettant au propriétaire d'un fichier d'en protéger le contenu en définissant des droits d'accès :

- En lecture (r)
- En écriture (w)
- En exécution ou en accès (x)

Lorsque l'on crée un répertoire par défaut il est : dwxr-xr-x c'est-à-dire d pour les répertoires et pour les fichiers -rwxr--r--
wxr pour donner les droits de écriture exécuter lecture
pour afficher page par page ls -al|more

De plus UNIX fonctionne en mode multi-utilisateur, c'est à dire qu'UNIX permet le partage de fichier au sein d'un groupe de travail ou bien même autorise le partage avec tout autre utilisateur en sachant que le propriétaire peut restreindre l'accès de ses derniers à ses fichiers.

La commande ls -al

Pour savoir les droits d'accès actuels appliqués à mes fichiers, j'utilise *ls-al*
cette commande me donne les caractéristiques suivantes :

- type de fichier :-,d,l,c,p,b
- autorisation d'accès
- nombre de sous répertoires
- identifiant du propriétaire
- groupe de travail du propriétaire
- taille du fichier (s'il s'agit d'un répertoire la taille est de 4096octets)
- date et heure de la dernière modification
- nom du fichier

La commande groups

La commande groups vralambo nous donne vralambo : asi01
Cette commande nous donne le nom du groupe auquel on appartient

Les alias

Pour créer ces alias, on a tapé la commande éditer le fichier caché .bashrc De cette manière ces alias seront effectifs à chaque connexion.

```
alias exe='chmod 100'
alias visible='chmod 704'
alias invisible='chmod 700'
alias deprotege='chmod 700'
alias protege='chmod 500'
```

La commande chmod

On peut changer les droits d'accès des fichiers, dont on est propriétaire. Pour cela on utilise la commande *chmod* .Les permissions accordées sont codées de 2 manières :

- En octal sur 3 octets, exemple : chmod 750 fichiers

$7=(111)_2$	$5=(101)_2$	$0=(000)_2$
r w x	r - x	- - -
propriétaire	Groupe de travail	autres

Par défaut on a 750 comme valeur pour un fichier ordinaire.

La commande umask

Par défaut le umask est 022, il définit les droits par défaut décrits ci dessus.
Si on fait umask 000, (000 est le complément de 777 pour chmod) cela donne tous les droits à tout le monde.
Si je tape 027 (complément 750) droits pas défaut définis ci dessus

Code de l'umask	Code numérique (au sens de "chmod")	Code (au sens de "ls -l")
0	7	rwx
1	6	rw-
2	5	r-x
3	4	r--
4	3	-wx
5	2	-w-
6	1	-x-
7	0	---

c - Les i-nodes

table des inodes
.

Les fichiers pointant vers un même objet ont le même inode.
Pour obtenir les inodes on utilise la commande ls –i

mchambre:
. 3975630
.. 3615987

TP7:
. 3827967
.. 3975630

Les liens

L'inode de mail.signature est 3974322 et celle de mail.signature1 est 3974783. Leurs n° inodes sont différents parce que ce sont des fichiers différents. il prend le début du n° d'inode du répertoire parent.

L'inode de bidon/mail.signature2 est 2144449.
L'indode de bidon/mail.signature est 3974322, la même que celle qu'il avait lorsqu'il était dans le répertoire compte.
Un fichier garde toujours la même inode quand il est déplacé, mais pas lors de sa copie.
mail.signature-bis est un fichier ordinaire, comme le fichier mail.signature avec le même accès et le même inode.

Remarque : La commande stat donne les caractéristiques suivantes

Nom de fichier *Type de fichier*
Taille
Droits d'accès : *Utilisateur (N° Id)* *Gr (N°Id)*
Place sur le disque *N°Inode* *Lien : Nb de fichier ayant cet Inode*
Dernier Accès
Dernière Modification
Dernier Changement

2 – Programmation Shell

a – Type de fichier

```
#!bin/bash
#typefile


echo -n "Repertoire courant =";pwd
repertoire=$(ls)

for f in $repertoire
      do
            if(test -d $f)
            then echo "$f est un répertoire"
            elif(test -f $f)
            then echo "$f est un fichier ordinaire"
            elif(test -b $f)
            then echo "$f est un fichier en mode bloc"
            elif(test -c $f)
            then echo "$f est un fichier en mode caractere"
            elif(test -p $f)
            then echo "$f est un ficher-tube"
            elif(test -s $f)
            then echo "$f est un ficher socket"
            fi
      done
```

La difficulté de ce script a été de voir comment affecter correctement des variables.La commande test permet de voir directement le type de fichier auquel on a affaire.

b – Les droits d'accès

```
#!bin/bash
# droits


if (test $# -eq 1)
then
if(test -e $1);

      then  echo "Fichier : $1"

      if(test -r $1)
                  then echo "vous avez les droits de lecture"
                  else echo "vous n'avez pas les droits de lecture"
      fi
      if(test -w $1)
                  then echo "vous avez les droits d'ecriture"
                  else echo "vous n'avez pas les droits d'ecriture"
      fi
      if(test -x $1)
                  then echo "vous avez les droits d'execution"
                  else echo "vous n'avez pas les droits d'execution"
      fi

      else echo "le fichier entré en paramètre n'existe pas"
fi
else echo "vous n'avez pas rentré le bon nombre de paramètres
vous devez donner le nom du fichier pour exécuter le script correctement"
fi
```

Pour ce script, on vérifie au préalable que le bon nombre de paramètres a été entré grace à $#

c – Les i-nodes

```
#!bin/bash
# inodefin

if(test -e $1);

      then  echo "Fichier : $1"
      temporaire=$(ls -i $1)
      inode=$(cut -d' ' -f2 $temporaire)
      echo " $inode"

else echo "le fichier n'existe pas"

fi
```

Lors de sa première exécution, ce script a fonctionné correctement, mais en voulant le réessayer pour un autre fichier, on a eu l'inode précédent avec :commande not found ...Après avoir cherché différentes solutions notamment mettre les résultats intermédiaires dans des fichiers, nous n'avons toujours pas compris pourquoi ce programme ne marche pas, et nous n'avons pas voulu recopier le corrigé on-line.

3 – Questions subsidiaires

- Comment UNIX gère-t-il les noms d'utilisateurs et de groupe (UID, GID) … où sont stockées tous ces informations.

UID

Définition

Le login name ou username est l'identifiant permettant de se connecter à un système UNIX. A un login name correspond un UID (valeur numérique) et un seul. Cette valeur est utilisé dans la gestion des droits d'accès. Cet association se fait dans le fichier /etc/passwd lors de la création de l'utilisateur

On attribue à chaque utilisateur un UID qui est typiquement un entier sur 16 ou 32 bits. Ce UID est contenu dans le fichier des mots de passe.

Les UID sont affectés par l'administrateur système.

Chaque processus qui s'exécute prend automatiquement l'UID de la personne qui l'a crée.
UID 0 est le super utilisateur ou root. Il a ce pouvoir de lire et d'écrire sur tous les fichiers systèmes quels que soit leur propriétaire et leur protection.

Pour pallier à une défaillance de la protection du système on associe un bit à chaque programme exécutable : le bit setUID, ce dernier se trouve dans le mot du mode de protection. Lorsqu'un programme dont le bit setUID est à 1 s'exécute, l'UID effectif du processus devient l'UID du propriétaire du fichier exécutable au lieu de l'UID de l'utilisateur qui l'invoque.

Comme pour les login name à un groupe name correspond une valeur numérique, celle ci est le GID. Le groupe name n'est jamais utilisé dans la gestion des droits, il n'est utilisé que pour une facilité d'administration. Une mauvaise administration des GID amène comme pour les UID des problèmes de cohérence. Pour éviter cet écueil il nécessaire de mettre en place une codification des noms de groupes et de leur associer de manière centralisé un GID

GID

Comme pour les login name à un groupe name correspond une valeur numérique, celle ci est le GID. Le groupe name n'est jamais utilisé dans la gestion des droits, il n'est utilisé que pour une facilité d'administration. Une mauvaise administration des GID amène comme pour les UID des problèmes de cohérence. Pour éviter cet écueil il nécessaire de mettre en place une codification des noms de groupes et de leur associer de manière centralisé un GID.

Source : Tanenbaum : systèmes d'exploitations

- Dans quel cas n'est-il pas possible d'utiliser de lien normal, mais un lien symbolique (ln –s)

Définition

Il y a 2 façons de représenter les noms des fichiers : les noms-symboliques à l'usage des utilisateurs et les structures internes (exemple un n° d'inodes)

Un lien symbolique est une entrée d'un répertoire qui s'interprète directement comme une chaîne de caractères. Cette chaîne sert à trouver le nom interne du fichier sur le serveur désigné.

Pour obtenir un lien physique, il faut que le lien figure sur le même système de fichier (et donc sur la même partition) que l'original. Dans le cas contraire, le numéro d'inode du fichier original ne correspond à rien, et il faut passer par un lien symbolique.

On peut comparer cela à la différence entre passage par valeurs et passage par adresse.