Introduction à l'informatique

L'arborescence

G. Santini, J.-C. Dubacq

IUT de Villetaneuse

S1 2016

Compléments sur l'arborescence

Propriété des fichiers

Identifications des utilisateurs dans un environnement multi-utilisateurs

UID (**U**ser **ID**entifier) numéro unique associé à chaque utilisateur lors de la création de son compte.

GID (**G**roup **ID**entifier) numéro unique d'un groupe d'utilisateurs. Chaque utilisateur peut être associé à un ou plusieurs groupes.

Utilité

- Chaque fichier (ou répertoire) et chaque processus du système est associé à un utilisateur : cet utilisateur est le propriétaire du fichier (ou répertoire) ou celui qui a lancé le processus.
- Être propriétaire d'un fichier ou d'un processus confère des droits sur ceux-ci.

Connaitre l'identité du propriétaire d'un processus ou d'un fichier

- Les commandes `top` et `ps` affichent le nom du propriétaire des processus.
- La commande `ls` avec l'option `-l` affiche le nom et le groupe du propriétaire d'un fichier ou d'un répertoire.
- Les UID et GID sont enregistrés dans le fichier d'administration `/etc/passwd` ou d'autres mécanismes

Syntaxe pour ls(ter)

```
ls -l <source>
```

Description

- Affiche le contenu d'un répertoire en format long.
- Le format long donne le nom du propriétaire et son groupe, ainsi que les droits des différentes classes d'utilisateurs sur les fichiers et répertoires.

Exemple d'utilisation:

chez_moi/ ..... Répertoire Courant
- public_html/
  - index.html
  - astronomie.txt

```
login@host:~$ ls -l
 total 32
 drwxr-xr-x 2 santini ensinfo 4096 20 jui 15:50 public_html
 -rw-r--r-- 1 santini ensinfo   25 20 jui 15:49 astronomie.txt
```

Ici, le nom de l'utilisateur est `santini`, nom du groupe est `ensinfo` et les droits sont colorés en vert.

Les droits sur les fichiers et répertoires

3 catégories d'utilisateurs

-	r w x	r w x	r w x
Type de Fichier	Doits du propriétaire (**U**ser)	Doits du groupe (**G**roup)	Doits des autres (**O**ther)

Types de fichiers

	Types
-	Fichier ordinaire
d	Répertoire
l	lien symbolique

Droits/Permissions

		Fichier	Répertoire
r	(**R**ead)	lire	lister le contenu
w	(**W**rite)	écrire et modifier	modifier le contenu
x	(e**X**ecute)	exécution	traverser

Types d'utilisateurs

		Cible
u	(**U**)ser	Propriétaire du fichier/répertoire
g	(**G**)roup	Membre du même groupe que le propriétaire
o	(**O**)ther	Tous les autres
a	(**A**)ll	Tous les utilisateurs (réunion de `'u'` `'g'` et `'o'`)

Syntaxe pour chmod

```
chmod  droit fichier
```

Description

- Modifie les droits et permissions accordés par le propriétaire aux différents utilisateurs du système.

Exemple d'utilisation:

Retire au propriétaire le droit d'écriture sur le fichier `cv_2011.pdf`.

```
login@host:~$ chmod u-w cv_2011.pdf
```

Ajoute au propriétaire et aux membres de son groupe le droit d'exécution sur le fichier `listing.bash`.

```
login@host:~$ chmod ug+x
listing.bash
```

Retire aux utilisateurs qui ne sont ni le propriétaire ni membre de son groupe les droits de lecture, d'écriture et d'exécution.

```
login@host:~$ chmod o-rwx
listing.bash
```

Ajoute à tous les utilisateurs, tous les droits.

```
login@host:~$ chmod a+rwx
listing.bash
```


Description

Il existe plusieurs notations des droits.

- La notation alphanumérique : `(ugoa) (+/-) (rwx)`
- La notation octale
  - Calcul des droits pour un utilisateur (u, g ou o) :

Droit	`---`	`--x`	`-w-`	`-wx`	`r--`	`r-x`	`rw-`	`rwx`
Binaire	000	001	010	011	100	101	110	111
Octale	0	1	2	3	4	5	6	7

  - Exemple de notation octale des droits d'un fichier

	User			**G**roup			**O**ther		
Alphabétique	r	w	x	r	-	x	-	-	x
Binaire	1	1	1	1	0	1	0	0	1
Octale	7			5			1		

Exemple d'utilisation:

Alph.	Oct.
`--- --- ---`	000
`rw- --- ---`	600
`rw- r-- r--`	644
`rw- rw- rw-`	666

Alph.	Oct.
`rwx --- ---`	700
`rwx r-x r-x`	755
`rwx rwx rwx`	777

```
login@host:~$ chmod 700 dir_parano
login@host:~$ chmod 644 fichier_pub
```


Exercices

Identification et droits

Q1 Au moyen de la commande `id`, affichez votre UID et votre GID ? Comparez-le avec celui de votre voisin de table. Qu'en concluez-vous ? Comparez-les avec celui de l'utilisateur `root`. Qu'en concluez-vous ?

Q2 Quels sont vos droits sur le répertoire racine `/`, `root`, `/tmp`, sur votre répertoire `~/`, et celui de votre voisin de table `~/../login_voisin`.

Q3 Pouvez-vous lire les données contenue dans le répertoire de votre voisin. Quelle commande permettrait de le faire ? Qui doit lancer la commande ?

Q4 Donnez les commandes octale et alphanumérique de changement de droits permettant :

- d'autoriser aux membres de votre groupe et aux "autres" l'accès en lecture aux images du répertoire `donnees_tdtp2/images`.
- de donnez les droits d'écriture aux membres de votre groupe sur le fichier `donnees_tdtp2/command_line.txt`
- de vous (le propriétaire) retirer toute possibilité de supprimer le fichier `donnees_tdtp2/0readme`

Q5 Imaginez comment donner à votre voisin un accès sous votre répertoire personnel à un répertoire dans lequel il aurait les droits d'écriture sur un fichier spécifique, que vous ne pourriez vous que lire (mais pas modifier). Il ne doit pas pouvoir créer un autre fichier chez vous. Comment faites vous pour effacer ce fichier ?

Exercices

Remise en état

Après toutes les modifications pouvant impliquer votre répertoire personnel, n'oubliez pas `chmod 711` pour remettre les modes de votre répertoire à leur état d'origine.

Les principaux répertoires et leur contenu

Répertoire	Contenu
/	Répertoire racine : Toutes les données accessibles par le système
/bin	Binaires exécutables des commandes de bases (cd, ls, mkdir, …)
/dev	Fichiers spéciaux correspondants aux périphériques
/etc	Fichiers de configuration (profile, passwd,fstab...)
/home	Les répertoires personnels des utilisateurs
/lib	Librairies partagées et modules du noyeau
/mnt	Points de montage des périphériques
/root	Répertoire personnel de l'administrateur
/tmp	Données temporaires
/usr	Ressources accessibles par les utilisateurs
/var	Fichiers de log ou fichiers changeant fréquemment

L'essentiel est synthétisé dans
`https://fr.wikipedia.org/wiki/Filesystem_Hierarchy_Standard`

Les principaux répertoires et leur contenu

Une structure plus ou moins normalisée

- Les fichiers nécessaires au fonctionnement du système sont organisés en arborescence,
- Cette arborescence est commune à presque toutes les distribution linux,
- Cette organisation rationalisée facilite l'installation de nouveaux programmes qui savent où trouver les fichiers dont ils peuvent avoir besoin.

Une organisation qui permet un cloisonnement

- Les fichiers et les répertoires systèmes sont protégés par des restrictions de droits,
- De nombreux fichiers ne peuvent être modifiés par un utilisateur « normal »,
- Seul l'utilisateur `root`, ou les utilisateur faisant partie du groupe `admin` peuvent avoir la permission de modifier certains fichiers.
- Il s'agit d'une protection. Pour réaliser une action susceptible d'affecter le comportement du système il faut montrer "patte blanche" et prendre conscience de ce que l'on fait. Entrer le mot de passe `root` doit être un signal d'alerte.

Exercices

Hiérarchie du système

Astuce : si la sortie d'une commande est trop longue, on peut ajouter `|less` à la fin de la ligne pour l'afficher par morceaux. Ceci vous sera expliqué dans quelques séances...

Q6 Identifiez le propriétaire, le groupe et les différents droits des fichiers contenus dans le répertoire `/bin` ? Quels sont vos droits sur ces fichiers ?

Q7 Ces fichiers on le droit `x`. Que pouvez-vous en conclure ?

Q8 A votre avis, que se passe-t-il en fait lorsque vous saisissez une commande telle que `ls` ?

FHS

Q9 Identifiez, à l'aide de la FHS, la fonction de `/usr/include`. Confirmez votre hypothèse en regardant quelques fichiers.

Langages Compilés Vs Langages Interprétés

Caractéristiques des Langages Compilés

- L'ensemble du code source est compilé une seule fois avant l'exécution en instructions machine (contenues dans un fichier : exécutable).
- Le compilateur n'est pas nécessaire lors de l'exécution.
- Le compilateur est spécifique à la machine.
- L'exécutable (code compilé) est spécifique à la machine.

Inconvenients

- Il faut recompiler pour prendre en compte une modification du code.
- L'exécutable n'est pas portable sur d'autres machines.

Avantages

- Plus rapide (spécifique à la machine qui exécute les instructions).
- L'ensemble des instructions sont regroupées dans un seul fichier.

Exemples de langages CompilésInterprétés

- C, C++, ADA, Pascal, Fortran, Cobol,

Langages Compilés Vs Langages Interprétés

Caractéristiques des Langages Interprétes

- Les instructions du code source sont converties en instructions machine lors de l'exécution du programme
- L'interpréteur est nécessaire lors de l'exécution.
- L'interpréteur est spécifique à la machine,
- L'exécutable (le code source) n'est pas spécifique à la machine.

Inconvenients

- Moins rapide.
- Plusieurs fichiers (et librairies) servent à l'exécution.

Avantages

- Modifications du code source immédiatement prises en compte lors de la réexécution.
- Le code est portable sur d'autres machine

Exemples de langages CompilésInterprétés

- Java, Python, Bash, Lisp, PHP, Prolog, Perl, Javascript

Lancer un programme/une commande

Cas général

- Pour exécuter un programme il suffit saisir sur la ligne de commande le chemin menant au fichier contenant les instructions,
- Si le fichier présente la permission `"X"` pour exécutable, les instructions qu'il contient sont exécutées.

Script bash exécutable

- Un script `bash` est un fichier texte contenant des instructions `bash`
- La première ligne contient le chemin menant à l'exécutable de l'interpréter précédé des caractère `#!` (par exemple `#! /bin/bash`),
- La seconde ligne est souvent vide,
- Les lignes suivantes comportent des instructions.

test_bash.sh

```
#!/bin/bash

instruction 1;
instruction 2;
...
instruction N;
```


Exercices

Lancer un programme/une commande

Q10 Après avoir créé un répertoire `bin` dans votre répertoire personnel, définissez créez dans ce répertoire un script nommé `listintro.sh`. Ce script comporte une unique commande permettant de lister le contenu du répertoire de travail `Intro_Systeme` dans lequel vous avez l'habitude de travailler.

Q11 Attribuez les droits d'exécution sur ce fichier. Il est normalement devenu un exécutable.

Q12 Quelle commande permet d'exécuter ce script si le répertoire courant es le répertoire `~/bin` qui le contient ? Idem, si le répertoire courant est votre répertoire personnel. Vous vérifierez que le script se comporte comme attendu (il vous place dans une autre répertoire).

Q13 la commande `echo` permet d'afficher une message à l'écran. Modifiez le script pour qu'il avertisse l'utilisateur de la fin du script par un message explicite.

Syntaxe pour echo

```
echo  expression
```

Description

- Affiche sur la sortie standard l'expression après interprétation.

Exemple d'utilisation:

Affiche `'Bonjour'` :

```
login@host:~$ echo Bonjour
Bonjour
login@host:~$
```

Définie une variable puis affiche sa valeur :

```
login@host:~$ Astre=Terre
login@host:~$ echo $Astre
Terre
login@host:~$ echo La planete $Astre
La planete Terre
login@host:~$
```


Lancer un programme/une commande

Cas général

- Pour exécuter un programme il suffit saisir sur la ligne de commande le chemin menant au fichier contenant les instructions,
- Si le fichier présente la permission `"X"` pour exécutable, les instructions qu'il contient sont exécutées.

Cas particulier : les commandes

- Une commande (`ls`, `gedit`, `firefox`, …) est un programme comme un autre,
- Les instructions qui doivent être évaluées sont écrites dans un fichier (`/bin/ls`, `/usr/bin/python`, `/usr/share/bin/firefox`, …),
- Pourtant …

Des chemins qui mènent nulle part ! ! !

- les noms des commandes (`ls`, `gedit`, `firefox` …) sont toujours saisies comme des chemins relatifs (pas de `/bin/...` devant le nom du fichier), alors que le fichier de commande n'est pas dans le répertoire courant ! …
- On donne donc un chemin vers un fichier qui n'existe pas …

Chemins par défaut et variable d'environnement

Lorsqu'on donne une commande au terminal, on ne spécifie pas le chemin vers le fichier qui contient l'exécutable, on donne juste le nom du fichier…

```
login@host:~$ ls
Mes_Documents/ Etoiles/ astronomie.txt
cv.pdf
login@host:~$
```

…alors, comment le système trouve-t-il le fichier a exécuter correspondant à la commande ?…

Un mécanisme propre aux commandes

- Le premier mot tapé sur la ligne de commande est toujours interprétée comme le nom d'un fichier exécutable,
- Le système recherche donc dans une liste de répertoires contenant les exécutables si un fichier porte le nom de cette commande,
- Dès qu'il trouve dans ces répertoires un tel fichier, il l'exécute …

Chemins par défaut et variable d'environnement

Les variables d'environnement

- Comme les variables d'un script, les variables d'environnement sont associées à une valeur,
- De telles variables sont définies par le système d'exploitation pour son fonctionnement, ce sont les variables d'environnement,
- ces variables peuvent être utilisées par les programmes.

La variable d'environnement $PATH

- Sa valeur est une liste de répertoires séparés par le signe `':'`

```
PATH=repertoire1:repertoire2:...:RepertoireN
```

- Lors de chaque appel de commande, l'interpréteur parcourt cette liste dans l'ordre à la recherche d'un fichier portant le nom de la commande,
- Dès qu'il rencontre un tel fichier, il met fin à sa recherche et exécute le fichier.

Rôle de $PATH

→ Il s'agit d'une liste de répertoires que l'interpréteur parcours automatiquement et séquentiellement (par défaut) si aucun chemin n'est donné pour trouver le fichier exécutable.

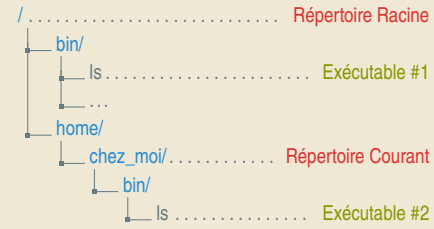
Syntaxe pour which

```
which  nom_de_la_commande
```

Description

- Affiche le chemin du fichier correspondant à une commande.
- Parcours successivement les répertoires de la variable $PATH. Dès qu'il trouve un fichier correspondant au nom de la commande il renvoie son chemin.

Exemple d'utilisation:




```
login@host:/home/chez_mo$ echo $PATH
/bin:/usr/bin:/usr/local/bin:/home/chez_moi
login@host:/home/chez_moi$ which ls
/bin/ls
```


Chemins par défaut et variable d'environnement

La commande `export` pour modifer la variable $PATH

- Définir la variable $PATH

```
login@host:~/$ export PATH=monDir1:monDir2
```

- Ajouter un répertoire à $PATH

```
login@host:~/$ export PATH=$PATH:monDir2
```


Exercices

Environnement

Q14 Au moyen de la commande `env`, donnez la liste des répertoires contenus dans $PATH.

Q15 Au moyen de la commande `which`, afficher la localisation des exécutables correspondants aux commandes `mv`, `cd`, `man`, `cat`, `firefox`, `acroread`.

Q16 Vérifiez que ces répertoires font partie de la liste contenue dans la variable $PATH ? Que se passerait-il si ce n'était pas le cas ?

Q17 Ajouter le répertoire `~/bin` à la liste des répertoires $PATH.

Q18 Maintenant que `~/bin` est parcoure par défaut lors de l'appel d'une commande, comment invoque-t-on désormais l'exécution du script `listintro.sh` ? Vérifiez le comportement attendu.

Fichiers de configuration

Fichiers systèmes et utilisateurs

- Les variables d'environnement (et d'autres variables de configuration) sont définis dans divers fichiers.
- On distingue les fichiers système qui définissent des comportements pour tous les utilisateurs (stockés dans le répertoire `/etc/`) des fichiers de configuration propres à un utilisateur (stockés dans le répertoire personnel)

fichier	Propriétaire	Applicable à	Évalué lors
`/etc/profile`	`root`	Tous	Au début de chaque shell de login
`/home/chez_moi/.profile`	`utilisateur`	utilisateur	Au début de chaque shell de loginl
`/etc/bashrc`	`root`	Tous	Au début de chaque shell
`/home/chez_moi/.bashrc`	`utilisateur`	utilisateur	Au début de chaque shell

Configurer son environnement

- Chaque utilisateur peut redéfinir ses variables d'environnement,
- Pour cela il peut modifier le contenu des fichiers `.bashrc` et `.profile` dans son répertoire personnel,
- Ce sont des fichiers cachés (leur nom commence par un point : `.`). Pour voir si ils existent il faut utiliser l'option `-a` de la commande `ls`

Fichiers de configuration

Contenu d'un fichier `.bashrc`

- Redéfinition des variables d'environnement,
- Définition des alias,
- Définition des fonctions,
- et de façon générale toutes les instructions que l'on souhaite évaluer lors de l'ouverture d'un nouveau shell.

.bashrc

```
# Mes aliases
alias ll='ls -l'
alias df='df -h'
alias rm='rm -i'
# Mes variables
PATH=$PATH:$HOME/bin
```

Autres variables d'environnement

`$HOME` le chemin du répertoire personnel de l'utilisateur,

`$PWD` le chemin du répertoire courant.

Syntaxe pour alias

```
alias  nom_de_la_commande=expression
```

Description

- créet un alias entre un nom de commande et une expression.
- l'expression est donnée entre *quotes* : *'expression ...'*

Exemple d'utilisation:

```
chez_moi/ ..... Répertoire
├── public_html/
│   └── index.html
└── astronomie.txt
```

```
login@host:~$ ll
-bash: ll: command not found
login@host:~$ alias ll='ls -l'
login@host:~$ ls -l
 total 32
 drwxr-xr-x 2 santini ensinfo 4096 20 jui 15:50     public_html
   -rw-r-r- 1 santini ensinfo   25 20 jui 15:49 telluriques.txt
```


Exercices

Chemins par défaut et variable d'environnement

Q19 Copiez l'exécutable de la commande `ls` dans le répertoire `~/bin`. Deux versions de la même commande existe dans 2 répertoires différents listés sans $PATH. Quelle commande est exécutée ? Comment en être sur et pourquoi ?

Q20 Si vous modifiez la variable $PATH, de la façon suivante, quelle commande est alors exécutée ?

```
login@host:~/$ export PATH=monDir2:$PATH
```

Q21 Modifiez/créez un fichier `~/.bashrc` pour ajouter le répertoire `~/bin` de façon stable à votre variable $PATH.

Q22 ajoutez dans le même fichier les alias qui vous paraissent intéressants.