

### Le système UNIX

#### Presentation du système UNIX

Yves STADLER

Codasystem, UPV-M

11 septembre 2011

1/30

#### Nous allons voir

- Le système de gestion de fichier d'UNIX
- Les bases du langage BASH

#### Notions abordées dans ce module

- L'interface CLI (Command Line Interface)
- Fichiers (types, droits, etc...)
- Commandes simples et paramétrées
- Programmes de commandes, scripts
- Interaction avec le système d'exploitation

2/30

## Historique et évolution

### Naissance

- UNIX est né en 1969, et conçu par Ken Thomson aux laboratoires Bell d'AT&T.
- Deux familles : BSD (Berkley Software Distribution) et SystemV (AT&T).

### Évolution

- Standardisation POSIX ← convergence des deux familles.
- Développement des IHM ← apparition des deux variantes : MOTIF (IBM) et OPEN LOOK (Sun et AT&T)

### Distribution

- Versions commerciales (Solaris et AIX)
- Domaine public avec Linux

3/30

## Caractéristiques

### Généralités

- Multi-utilisateurs
- Multi-tâches
- Universel
- SGF (Système de gestion de fichier) arborescent
- Processus hiérarchisés

4/30

## Portabilité

L'objectif d'UNIX est d'avoir une conception portable (i.e. compatible et adaptable)

## Composantes

- Noyau
- Interface homme-machine
- Outils

5/30

## Définition

Activité associée à l'exécution d'un programme.

## Gestion

- Interface assurée par une bibliothèque de fonctions primitives du langage C
- Création de processus : `fork()`
- Synchronisation et communications avec d'autres processus : `wait()`, `pipe()`, ...
- Relation hiérarchisée des processus : père-fils.

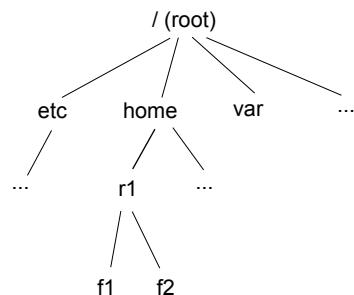
6/30

# Système de gestion de fichiers

## La structure arborescente

- Structure arborescente = répertoire (directory)
- Les noeuds sont des répertoires
- Les feuilles sont des fichiers
- La racine est le point de départ de la structure (en anglais : root)

title



7/30

# Système de gestion de fichiers

## Exemples de répertoires

- dev : concerne les périphériques
- bin : contient des exécutables
- etc : fichiers pour les utilisateurs systèmes
- usr : répertoire des programmes utilisateurs, `/usr/bin` commandes du systèmes

8/30

### Deux types d'accès

Accéder à l'arborescence, c'est désigner la référence (fichier ou répertoire) que l'on souhaite utiliser :

- par un chemin
- par un lien

### Chemin arborescent

Deux manières :

- par un chemin absolu
- par un chemin relatif

### Exemple avec le fichier f1

- Absolu : `/home/r1/f1`
- Si l'on se trouve dans le répertoire `/home` : `r1/f1`

### Lien

On peut créer des liens désignant un fichier, ou un répertoire (commande `ln`)

- Ils apparaissent comme un fichier, ou un répertoire (transparence)
- Ils ont les mêmes droits
- Ils accèdent aux même emplacement physique

### Remarques

Modifier le contenu d'un lien modifie le contenu de l'original, mais détruire un lien ne détruit pas l'original.

### Trois types

- Ordinaire (suite d'octets structurés selon un format)
- Répertoire (contient les références à d'autres fichiers, et leurs attributs : droits, position sur le disque, ...)
- Spécial (associé à un périphérique)

## Protection des fichiers

### Type de protection

- Contre la lecture
- Contre la modification
- Contre l'exécution

### Interdire qui ?

- Moi
- Mon groupe
- Les autres

13/30

## Protection des fichiers

### Champs (attribut de fichier)

U	G	O
r x w	r x w	r x w

### Légende

- User Group Others
- Read Write eXecute
- Absence de droit : -

14/30

## Protection des fichiers

### Change mode

```
chmod 777 [fich1] [fich2]
chmod u+rxw [fich1] [fich2]
```

### Correspondance

U	G	O
1 1 1 = 7	0 0 1 = 1	0 0 0 = 0
r x w	r x w	r x w

15/30

## Langage de commande

### CLI

Ou Command Line Interface, c'est ce qui permet à l'utilisateur par le moyen de mots clés, d'agir directement avec le système.

### Interpréteur de commande

Appelé aussi terminal il constitue l'interface entre l'utilisateur et la CLI.

- Il permet de se loguer
- Il permet de saisir des commandes (entrée standard)
- Il permet d'afficher les résultats (sortie standard)
- Il permet de voir les erreurs (sortie d'erreur standard)

16/30

## Commandes de base

### Juste après le login

- `whoami` : me dit qui je suis (ex : ystadler)
- `id` : mon numéro d'utilisateur (ex : uid=500)
- `groups` : à quel groupe j'appartiens (ex : profs)
- `pwd` : dans quel répertoire je suis

### Les répertoires

- `cd [directory]` : change directory
- `cd ..` : revenir au répertoire parent
- `cd .` : changer pour le répertoire courant
- `~` : home, répertoire de l'utilisateur en cours

17/30

## Commandes de bases

### List

- `ls [directory]` : donne la liste des répertoires et fichiers
- `ls -a` : liste tous les fichiers/rep
- `ls -l` : affiche une liste de fichiers/rep avec leurs attributs

### ls -l

```
drwx--r-x 2 root root 47 Jun 10 2008 Desktop/  
drwx--r-x 2 root root 12 Jun 10 2008 F1/  
-rxw--r-x 2 root root 121 Jun 10 2008 f1  
-rxw--r-x 2 root root 113 Jun 10 2008 f2  
-rxw--r-x 2 root root 114 Jun 10 2008 f3
```

18/30

## Commandes de base

### Lecture, écriture

- `cat [fichier]` : liste le contenu d'un fichier sur l'entrée standard
- `cat` : on lit sur stdin et on écrit sur stdout
- `cat > fichier` : écrit dans un fichier
- CTRL-D = End Of File (EOF)

### Création, destruction

- `mkdir directory` : créer un répertoire
- `touch file` : créer un fichier vide (ou modifie le last accès d'un fichier existant)
- `rmdir directory` : supprimer un répertoire
- `rm file` : supprimer un fichier

19/30

## Commandes de bases

### Les options

- `rm -f file` : supprimer un fichier sans confirmation
- `rm -r` : suppression récursive
- `rm -rf` : suppression récursive sans confirmation
- `mkdir -p ../../directory` : créer un répertoire et ses parents si nécessaire.

### Les options longues

- `rm -help` : affiche l'aide
- `./configure --prefix=/usr/bin`

20/30

## Find et grep

### Find

```
find [path] [expression]
expression := (EXPR) | !EXPR | -not EXPR | EXPR1 -a(nd) EXPR2 |
EXPR1 -o(r) EXPR2
```

### grep

grep [-iv] terme [fichier] : recherche dans les fichiers le terme (-i cas insensitive, -v negation)

### Méta-caractères

- ^ début de ligne
- \$ fin de ligne

21/30

## Les scripts

### Définition

Liste de commande que l'interpréteur de commande va interpréter. C'est un programme.

### Le Sha bang

#!/bin/bash : il indique le type d'interpréteur que l'on va utiliser pour exécuter le programme.

#!/usr/bin/python

### Remarque

Un script doit être exécutable pour pouvoir se lancer de cette manière :  
./script ; Sinon il faut appeler l'interpréteur soit même : bash script

22/30

## Variables

### Les variables

- Affectation : a=valeur
- Lecture : \$a

### Variable d'environnement

- Variables utiles pour les processus systèmes
- On les listes avec : env
- \$PWD : répertoire en cours
- \$PATH : chemins d'accès des commandes exécutables directement.
- \$SHELL : shell utilisé par défaut.

23/30

## Tests

### If... then... else.. fi

```
if (test); then (command); else (command); fi
if
(test)
then
(command)
else
(command)
fi
```

### Exemple

```
if
[ $a -eq 2 ]
then
echo "Hello"
else
echo "World"
fi
```

24/30

### Faire un test

- [ expression ]
- -a fichier : vrai si le fichier existe
- -s fichier : vrai si le fichier existe, et a une taille non nulle.
- -d fichier : vrai si le fichier est un répertoire
- -r fichier : vrai si le fichier est exécutable (idem pour -w -x)
- -z chaîne : vrai si la chaîne est nulle
- -n chaîne : vrai si la longueur de la chaîne est non-nulle.

25/30

### Faire un test

- chaîne\_1 == chaîne\_2 Vrai si les deux chaînes sont égales. Le symbole = peut servir à remplacer ==
- chaîne\_1 != chaîne\_2 Vrai si les deux chaînes sont différentes.
- chaîne\_1 < chaîne\_2 Vrai si chaîne\_1 se trouve avant chaîne\_2 dans l'ordre lexicographique de la localisation en cours.
- chaîne\_1 > chaîne\_2 Vrai si chaîne\_1 se trouve après chaîne\_2 dans l'ordre lexicographique de la localisation en cours.
- arg1 OP arg2 OP est l'un des opérateurs suivants -eq, -ne, -lt, -le, -gt, ou -ge.

26/30

## Boucles

### For in do done

```
for (variable) in (liste de valeur)
do
(commands)
done
```

### while do done

```
while/until (test)
do
(commands)
done
```

27/30

## Case et select

### case

```
case (variable) in
valeur) (commands);;
valeur2) (commands);;
*) (commands);;
esac
```

### select

```
select (variable) in (liste)
Créer un menu numéroté pour chaque entrée de la liste
La selection de l'utilisateur est stocké dans la variable.
```

28/30

### Quelques variables symboliques

- \$0 : nom de la commande invoquée
- \$n : valeur du nième argument (1-9)
- la commande shift permet de décaler les arguments (2 dans 1, 3 dans 2, etc...)
- \$# : le nombre d'arguments utilisés lors de l'appel du script

La suite en TD.