Le système UNIX

Presentation du système UNIX

Yves Stadler

Codasystem, UPV-M

13 mars 2013

Objectifs

Nous allons voir

- Le système de gestion de fichier d'Unix
- Les bases du langages Bash

Notions abordées dans ce module

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

Historique et évolutions

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)

Distributions

- Versions commerciales (Mac OS X, Hp-ux, . . .)
- Versions libre (*Bsd, Linux, . . .)

Caractéristiques

Généralités

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

Universalité

Portabilité

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

Composantes

- Noyau
- Interface homme-machine
- Outils

Gestion des processus

Définition

Activité associée a l'exécution d'un programme.

Gestion

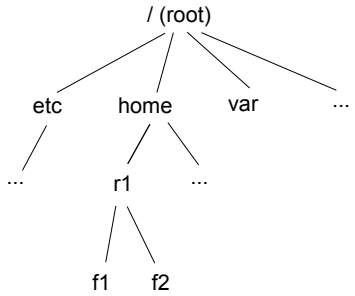
- Interface assurée par une bibliothèque de fonctions primitives du langage C
- Création de processus : fork()
- Synchronisation et communications avec d'autres processus : wait(), pipe(), ...
- Relation hiérarchisée des processus : père-fils.

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



Système de gestion de fichiers

Exemples de répertoires

- dev : concerne les périphériques
- bin : contient les exécutables essentiels au système
- etc : fichiers de configuration
- usr : répertoire des programmes utilisateurs

Accès à l'arborescence

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

Accès à l'arborescence

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`

Accès à l'arborescence

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.

Types de fichiers

Trois types

- Ordinaire (suite d'octets stucturée 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

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 : -

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

Langage de commande

Interpréteur de commande

Appellé aussi Shell il constitue l'interface entre l'utilisateur et le système grâce à un langage simple et l'accès à des utilitaires appelés coreutils.

- 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)

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
- `cd -` : revenir au répertoire précédent
- Voir aussi `pushd` et `popd`
- ~ : home, répertoire de l'utilisateur en cours

Commandes de bases

List

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

ls -l

```
drxw--r-x 2 root root 47 Jun 10 2008 Desktop/
drxw--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
```


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

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`

Find et grep

Find

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

grep

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

Méta-caractères

- ^début de ligne
- $ fin de ligne

Les scripts

Définition

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

Le Shebang

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

Remarque

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

Variables

Les variables

- Affectation : a=valeur
- Lecture : $a

Variable d'environnement

- Variables utiles pour les processus systèmes
- On les listes avec : `env`
- $USER : utilisateur courant
- $HOME : dossier de l'utilisateur courant
- $PATH : chemins d'accès des commandes exécutables directement.
- $SHELL : shell par défaut de l'utilisateur.
- $PWD : répertoire en cours

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
```

Les tests

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 chaine : vrai si la chaine est nulle
- -n chaîne : vrai si la longueur de la chaîne est non-nulle.

Les tests

Faire un test

- chaîne_1 = chaîne_2 Vrai si les deux chaînes sont égales.
- 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.

Boucles

For in do done

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

while do done

while/until (test)
do
(commands)
done

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.

Variables utiles

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

Merci

La suite en TD.