- DEVOPS ORIENTÉE OPS

Votre formateur

Loïc Guillois

Développeur Web full stack depuis plus de 10 ans

Expérience en environnement

- SSII
- Grand comptes Banque / Assurance / La Poste
- Startup (Gamific.TV, La Fourchette.com, Akeneo)
- Indépendant / freelance

Enseignant / formateur depuis 5 ans

- Développement
- No SQL
- Devops

Objectifs pédagogiques

- Acquérir la connaissance des commandes fondamentales des systèmes d'exploitation Unix et Linux à travers des exercices modulaires de difficulté progressive
- Devenir autonome pour une première prise en main d'un système
- Passer l'étape importante de la maîtrise de l'éditeur vi

Introduction

Loïc Guillois

Historique

Système d'exploitation :

Programme de base qui assure la gestion du matériel (clavier, écran, disque dur...), du système de fichier et des applications des utilisateurs

Deux grandes familles:

- Windows : XP, Vista, Seven, 10…
- Unix : Solaris, *BSD, Linux, MacOS...

Historique

L’origine d’Unix :

Unix a été conçu en 1969 aux Bell Labs (AT&T) par des ingénieurs puis réécrit en langage C puis porté sur de nombreuses architectures matérielles avec une importante contribution de l’université de Berkeley.

Historique

Unix, définition :

Unix est une famille de systèmes d'exploitation multitâche et multi-utilisateur dérivé du Unix d'origine créé par AT&T

Il repose sur un interpréteur (le shell) et de nombreux petits utilitaires, accomplissant chacun une action spécifique et appelés depuis la ligne de commande.

Il s’appuie sur des standards : POSIX ou System V

Historique

L’origine de Linux :

En 1991 un étudiant finlandais, Linus Torvalds, décida de concevoir un système d'exploitation capable de fonctionner sur les architectures à base de processeur Intel 80386.

Linux ne contient aucun code provenant de Unix, il en est juste inspiré, et complètement réécrit. D'autre part, Linux est un logiciel libre.

Caractéristiques

Un outil, une fonctionnalité :

Unix a initialement été conçu pour disposer de nombreux petits programmes, chacun effectuant un nombre limité de tâches, le plus souvent une seule

Ils sont utilisés en cascade (pipe) pour accomplir des tâches plus complexes

Caractéristiques

Tout est fichier :

Sous Unix, “tout est fichier”. Cela revient à dire que pour manipuler le matériel d’un ordinateur on utilisera des opérations de lecture écriture/classique pour la collecte d’information et l’envoi d’information.

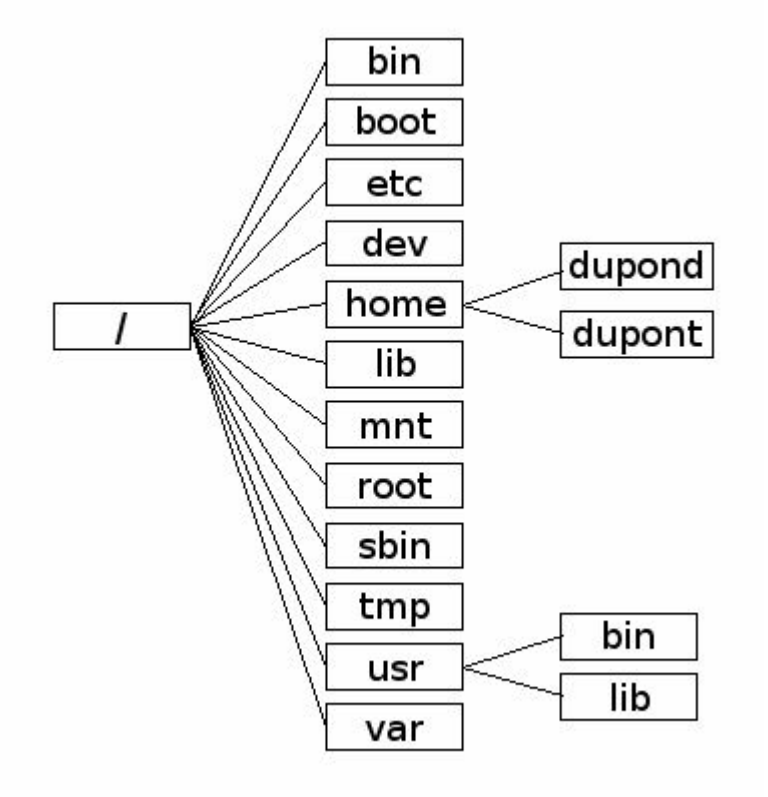
On écrira dans un fichier spécifique pour lire un fichier audio

On lira dans un fichier spécifique pour récupérer les informations sur le processeur

etc.

Système de fichiers

Système de fichier de type arborescence :



Système de fichiers

Système de fichier de type arborescence :

- La racine est /
- Il n'y a pas plusieurs "disques" comme sous Windows avec C:/ D:/ etc.
- Il est sensible à la casse: Monfichier et monfichier sont deux fichiers différents
- Les espaces et caractères accentués dans les noms de fichiers et répertoires sont à proscrire

Système de fichiers

Système de fichier de type arborescence :

- Ext2, Ext3, Ext4
- ReiserFS
- JFS
- XFS
- …

Linux support les systèmes de fichier Windows (FAT, FAT32, NTFS)

Ces systèmes ont des limites de tailles de fichier, de taille de partition, un système de journalisation ou non, une gestion des droits ou non. Ils peuvent avoir des performances différentes selon l'utilisation (lecture/écriture, petits ou gros fichiers)

Système de fichiers

Système de fichier de type arborescence :

Essayez de trouver le système de fichier installé sur votre ordinateur

Utilisez les outils graphiques à votre dispositions, nous verrons plus tard comment le faire en ligne de commande

Débuter avec le Shell

Le shell :

Les systèmes Unix disposent d'un grand nombre d'interpréteurs de commandes, appelés shells Unix. On peut notamment citer sh, bash, ksh, tcsh ou encore zsh

Le plus courant est bash

Ouvrez un terminal et tappez la commande suivante :

```
echo "hello world"
```

Une session

Loïc Guillois

Connexion

Il faut s'identifier pour avoir accès à un ordinateur sous Unix avec un couple utilisateur / mot de passe

Pourquoi?

- Système multi-utilisateurs: accéder à son environnement
- Sécurité: pas d'utilisateur non autorisé.
- Confidentialité: dupont ne lit le fichier de dubreuil que si dubreuil le décide (système de droits)

Connexion

L'utilisateur “root”

C'est l'administrateur. Certaines opérations ne peuvent être faites que pas lui. Il peut tout faire et c’est potentiellement dangereux

On n’utilise l’utilisateur “root” que rarement et avec parcimonie

Interface graphique

Serveur X :

Tous les UNIX utilisent habituellement le même environnement graphique. Celui-ci a été développé au Massachusetts Institute of Technology (MIT) en 1984 et il s'appelle X. On peut aussi dire X11 ou X Window System.

Windows est né un an plus tard, pour servir d'environnement graphique au système d'exploitation MS-DOS.

Interface graphique

Serveur X :

- XFree86
- X.org
- Accelerated X
- etc.

X.org est utilisé par Linux et FreeBSD. Sachez cependant que d'autres UNIX (AIX, Solaris, etc.) font plutôt appel à Accelerated X

Interface graphique

Environnements de bureau :

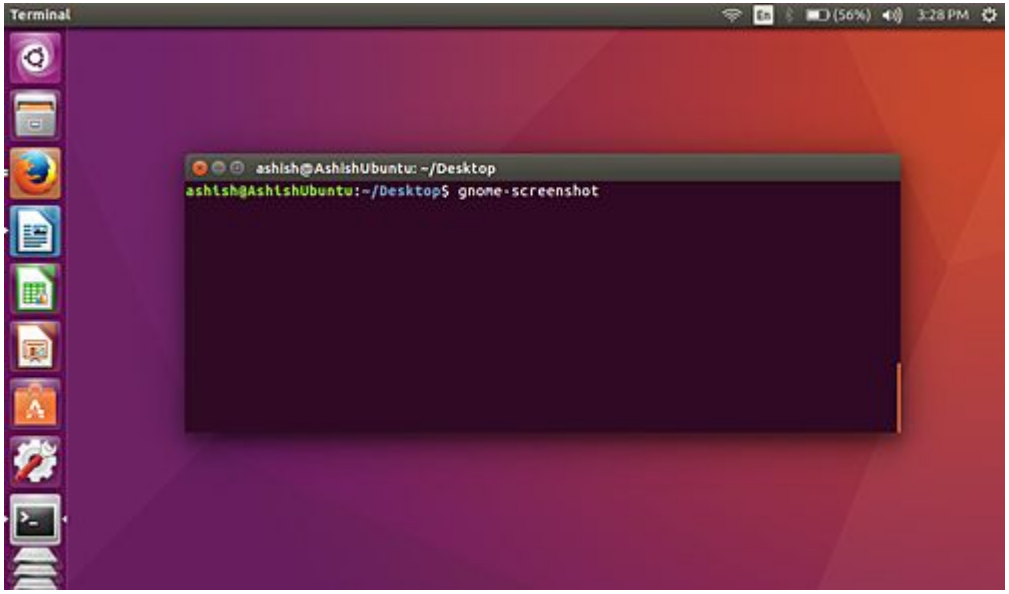
Le serveur X ne sert qu'à effectuer les opérations d'affichage de base. Il ne peut pas être utilisé seul. On adoptera un gestionnaire de bureau dont les plus courant sont :

- Gnome
- KDE
- XFCE
- ...

Ce sont eux qui donnent le style graphique des fenêtres, propose la gestion du fond d'écran ou encore les différents menu de l'interface utilisateur

Interface graphique

Exemple, Gnome (Ubuntu) :



L’aide avec man

Le manuel

La commande man xxx (pour manuel) permet d'afficher une documentation souvent très complète sur la commande xxx

Essayez donc: `man man`

- La commande man est utile, en particulier, pour explorer les options possibles d'une commande.
- Flèches ↑ et ↓, barre d'espace et touche b pour faire défiler le manuel.
- En fin de manuel (parfois) une liste de commandes sur le même thème.

Les commandes indispensables de l'éditeur Vi

Loïc Guillois

Editeur de texte vi

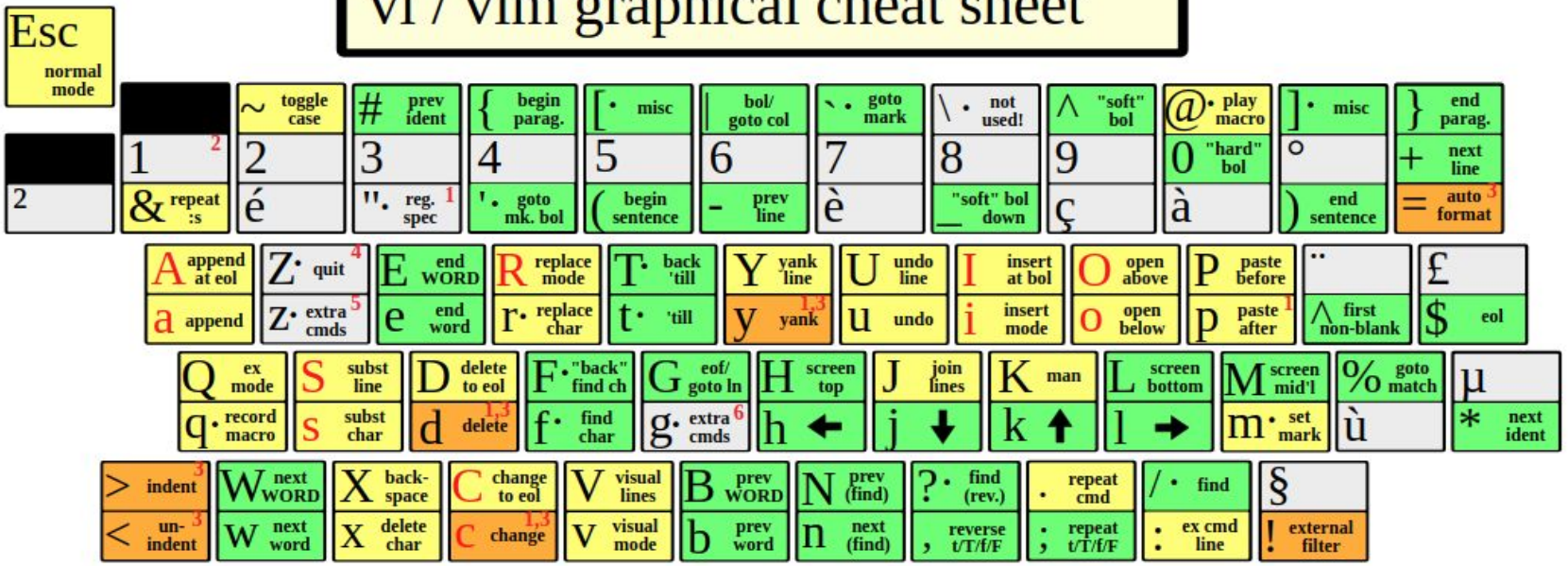
Vi :

Vi est un éditeur de texte en ligne de commande.

Il est le plus utilisé et parfois nous n'avons pas le choix que de l'utiliser et nous devons nous passer de l'interface graphique, par exemple sur un serveur distant ou sur une machine démarrée en "mode sans échec"

version 1.3
February 19th, 2013

vi / vim graphical cheat sheet



motion	moves the cursor, or defines the range for an operator
command	direct action command, if red , it enters insert mode
operator	requires a motion afterwards, operates between cursor & destination
extra	special functions, requires extra input
q·	commands with a dot need a char argument afterwards

bol = beginning of line, eol = end of line,
mk = mark, yank = copy

words: `quux(foo, bar, baz);`
WORDs: `quux(foo, bar, baz);`

Main command line commands ('ex'):

:w (save), :q (quit), :q! (quit w/o saving)
:e f (open file f),
:%s/x/y/g (replace 'x' by 'y' filewide),
:h (help in vim), :new (new file in vim),

Other important commands:

CTRL-R: redo (vim),
CTRL-F/-B: page up/down,
CTRL-E/-Y: scroll line up/down,
CTRL-V: block-visual mode (vim only)

Visual mode:

Move around and type operator to act on selected region (vim only)

Notes:

(1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z,*) (e.g.: "ay$ to copy rest of line to reg 'a')

(2) type in a number before any action to repeat it that number of times (e.g.: 2p, d2w, 5i, d4j)

(3) duplicate operator to act on current line (dd = delete line, >> = indent line)

(4) ZZ to save & quit, ZQ to quit w/o saving

(5) zt: scroll cursor to top, zb: bottom, zz: center

(6) gg: top of file (vim only), gf: open file under cursor (vim only)

AJC

VI - Cheat Sheet

Version: 1.0 / 12.06.2011
Nicole Cordes
http://www.cps-it.de

CPS-IT mehr wert im netz www.cps-it.de

Edit commands	
u	Undo last command
U	Undo all commands for current line
.	Repeat last command
y<MC>	Copy text depending on movement command
yy	Copy current line (yank)
5yy	Copy 5 lines from current
p	Paste (below)
P	Paste (above)
d<MC>	Delete text depending on movement command
D	Delete text to the end of line
dd	Delete current line
5dd	Delete 5 lines downwards from current
:10,20d	Delete lines 10 - 20
x	Delete current character
5x	Delete 5 characters from current
d0	Delete to beginning of line
d$	Delete to end of line

File commands	
:q	Quit
:q!	Quit without saving
:w	Save changes
:wq, :x, ZZ	Save and exit

Insert commands	
[ESC]	Switch back to command mode
i	Insert before cursor
I	Insert at the beginning of line
a	Insert after cursor
A	Insert at the end of line
c<MC>	Cut text depending on movement command
C	Cut text to the end of line
cc	Cut text of the current line
o	Insert new line (below)
O	Insert new line (above)

Movement commands <MC>	
l, [LEFT]	Move one character left (lower L)
5l	Move 5 characters left
h, [RIGHT]	Move one character right
5h	Move 5 characters right
k, [UP]	Move one line upwards
5k	Move 5 lines upwards
j, [DOWN]	Move one line downwards
5j	Move 5 lines downwards
w	Move one word right
5w	Move 5 words right
b	Move one word left
5b	Move 5 words left
e	Move to the end of current word
0	Move to beginning of line
^ _	Move to first non-blank character of line (underscore)
$	Move to end of line
+	Move to first character of next line
-	Move to first character of previous line
(	Move to next sentence
)	Move to previous sentence
]]	Move to next section
[[	Move to previous section
}	Move to next paragraph
{	Move to previous paragraph
%	Move to the corresponding opening/closing bracket (()'s, []'s and {}'s)
G	Go to last line
5G	Go to line 5

Miscellaneous	
~	Toogle upper / lower case
J	Join lines
:!ls	Run „ls" command from editor
:r foo.bar	Read file foo.bar into current file

Replace commands	
r	Replace current character
R	Enter replacement mode
:s/foo/bar	Replace next occurence of „foo" with „bar"
:s/foo/bar/g	Replace all occurences of „foo" with „bar"
:n,ms/foo/bar/g	Replace all occurences of „foo" with „bar" from line n to m
:%s/foo/bar/g	Replace all occurences of „foo" with „bar" in file
:%s/foo/bar/gc	Replace all occurences of „foo" with „bar" in file with confirmation

Screen commands	
z	Position line with cursor at top
z.	Position line with cursor at middle
z-	Position line with cursor at bottom
H	Go to top of screen (high)
5H	Go to 5th line from top of screen
M	Go to middle of screen (middle)
L	Go to bottom of screen (low)
5L	Go to 5th line from bottom of screen
v	Switch to visual mode (show selection)

Search commands	
fc	Search forwards for „c" (one character)
Fc	Search backwards for „c" (one character)
tc	Search forwards for „c" and go one character backwards
Tc	Search backwards for „c" and go one character forwards
;	Repeat search (keep direction)
,	Repeat search (reverse direction)
/foo	Search forwards for „foo"
?foo	Search backwards for „foo"
n	Repeat search (keep direction)
N	Repeat search (reverse direction)

Quelques commandes de base

Loïc Guillois

Quelques commandes en vrac

- `uname` : affiche le nom de la machine et quelques informations (version noyau…)
- `date` : affiche la date
- `watch`: permet d'exécuter une commande à interval régulier, exemple : watch date
- `cal`: affiche le calendrier
- `id` : affiche le nom de l'utilisateur courant
- `who` : affiche la liste des utilisateurs connectés
- `whoami` : affiche le nom de l'utilisateur courant
- `su` : permet de changer d'utilisateur
- `sudo` : permet d'exécuter une commande en tant que root
- `yes` : affiche à l'infini un message

Astuce : la touche tab permet l'autocomplétion des commandes et des noms de fichier

Les fichiers et les répertoires

Loïc Guillois

Les commandes de gestion des fichiers

Commandes de base :

- Afficher la liste des fichiers: `ls`

`ls` : affiche la liste des fichiers et sous-répertoire du répertoire courant

`ls rep1/toto` : affiche la liste des fichiers et sous-répertoires du répertoire rep1/toto

`ls -l` : affiche une liste détaillée (droits, propriétaire, taille, etc...)

`ls -a` : affiche également les fichiers cachés

`ls -t` : affiche par ordre de date de dernière modification

Les commandes de gestion des fichiers

Commandes de base :

- Mettre à jour la date de modification d'un fichier: `touch`

```
touch toto.txt
```

Le fichier sera créé avec les droits par défauts s'il n'existe pas.

Vous pouvez écrire dans un fichier hello avec la commande `vi`.

`cat hello` : permet de lire le contenu d'un fichier texte

`more hello`: permet de lire un fichier avec la possibilité de naviguer

Les commandes de gestion des fichiers

Commandes de base :

- Copier un fichier: `cp`

`cp fich1 fich2` : copie le fichier fich1 dans le fichier fich2 du répertoire courant.

`cp fich1 rep1/fich2` : copie du répertoire courant vers un sous-répertoire. cp -R rep1 rep2 : copie toute la arborescence de rep1 dans rep2.

- Déplacer, renommer un fichier: mv

`mv fich1 fich2` : renomme fich1 en fich2

`mv fich1 ../fich2` : déplace en le renommant le fichier fich1 vers le répertoire parent

Les commandes de gestion des fichiers

Commandes de base :

Détruire un fichier: `rm`

`rm fich1` : détruire le fichier fich1

`rm rep1/*` : détruit tout les fichiers dans rep1

`rm -f rep1/*` : même chose, sans demande de confirmation

`rm -rf rep1` : détruit récursivement rep1 et ses sous répertoires

Si root tape « rm -rf / » il détruit tout ce qu'il y a sur le disque

Les commandes de gestion des fichiers

Gestion de l'espace disque:

Espace disque occupé par un répertoire: `du -ks rep1` :

espace disque occupé par l'arborescence du répertoire rep1

`du -ks *` : espace disque occupé par chaque sous répertoire du répertoire courant.

Liste des partitions et espace occupé:

`df` : liste des partitions montées, dont /home, avec espace occupé/libre.

On peut aussi utiliser la commande `mount` pour lister les partitions montées sans informations sur l'espace occupé

Les commandes de gestion des dossiers

Se déplacer dans l'arborescence :

- Où suis-je? `pwd` : affiche le chemin absolu pour le répertoire courant
- Changer de répertoire: `cd rep1` : rentre dans le sous-répertoire rep1 du répertoire courant
- `cd /rep1`: tente d'entrer dans le répertoire de chemin absolu /rep1
- Créer un répertoire: `mkdir rep1` : crée le répertoire rep1 comme sous répertoire du répertoire courant

Les commandes de gestion des dossiers

Se déplacer dans l'arborescence :

- `..` : désigne le répertoire parent du répertoire courant
- `~` : désigne votre répertoire personnel /home/dupont par exemple
- `.` : désigne le répertoire courant.
- `-` : désigne le répertoire dans lequel on se trouvait précédemment

Utilisation: `cd ..` , `cd ~` (identique à cd sans argument), `cd ~/rep1`, `cd -`

La recherche de fichier

Plusieurs commandes de recherche :

`(s)locate toto.gif` : recherche toto.gif sur toute l'arborescence.
`find rep1 -name toto.gif` : recherche, récursivement, les fichiers nommés toto.gif dans le dossier et ses sous-dossier. (`man find`)
`which f90` : recherche l'exécutable f90 dans les répertoires de PATH

La commande `locate` est plus rapide que `find` car elle utilise un index de recherche, cependant celui ci doit être maintenu à jour pour être efficace

Les liens

Loïc Guillois

Concept

Liens symboliques :

Les liens symboliques sont des fichiers spéciaux qui ne font que pointer vers un autre fichier ou répertoire. C'est l'équivalent des raccourcis sous Windows.

Création de liens

Créer un lien symbolique :

Créer un lien symbolique: `ln -s cible nom_du_lien`

Cible peut être, par exemple, un chemin long à taper, dont on a souvent besoin.

Les liens symboliques

Repérer un lien symbolique :

```
ls -l /usr/local/bin
```

```
lrwxrwxrwx 1 root root           2 nov.  19 16:50  xzcat -> xz
```

Ici, les commandes xzcat et xz sont identiques

Les droits

Loïc Guillois

Les utilisateurs et groupes

Des limitations de droit :

```
lguillois@galibier:~$ cd /root
bash: cd: /root: Permission non accordée
```

Dans cet exemple, on remarque l'utilisateur n'a pas le droit d'accéder au dossier de l'utilisateur root. Ce qui est normal.

Un utilisateur a forcément des droits limités.

Les utilisateurs et groupes

Des limitations de droit :

Il existe 3 types de droits, applicables à 3 classes d'utilisateurs.

- r : droit de lire dans un fichier/répertoire.
- w : droit d'écrire dans un fichier/répertoire.
- x : droit d'exécuter un fichier/répertoire.

Les caractères 2,3 et 4 s'appliquent au propriétaire du fichier/répertoire, les 5,6 et 7 s'appliquent au groupe du propriétaire, les 8,9 et 10 s'appliquent à tout les autres utilisateurs.

Un tiret est affiché lorsque le droit n’est pas accordé

Les utilisateurs et groupes

Des limitations de droit :

La lecture, l'écriture et l'exécution se comprennent intuitivement pour les fichier.

Pour les dossiers, c'est un peu plus compliqué :

Pour faire `cd /home/dubreuil`, dupont doit avoir le droit d'exécution (x) sur le répertoire.
Pour faire `ls /home/dubreuil`, il doit avoir le droit de lecture (r).

Affichage et modification des droits

Afficher les droits :

`ls -la` : permet d'afficher les droits des fichiers du dossier courant

Modifier les droits:

On utilise la commande `chmod`

Affichage et modification des droits

Modifier les droits:

Trois symboles suivent chmod:

- 1 er symbole: u , change les droits du propriétaire g , change les droits du groupe du propriétaire o , change de tout les autres
- 2 nd symbole: + , ajoute un droit - , enlève un droit
- 3 ième symbole: r,w ou x…

Exemple:

`chmod o+x /home/dubreuil` : exécuté par dubreuil, donne le droit a tout le monde les droits d'exécution de son répertoire principal.

Droits par défaut

```
lguillois@galibier:/tmp$ touch titi
lguillois@galibier:/tmp$ ls -la titi
-rw-r--r-- 1 lguillois lguillois 0 déc.  30 22:55
titi
```

Les droits par défaut : droit en lecture pour tout le monde, droit en écriture pour le propriétaire.

Attention: même si les autres utilisateurs ont les droits sur un fichier, il faut d'abord qu'ils aient les droits sur les dossiers contenant pour interagir.

Gestion des utilisateurs

Il est possible de créer un utilisateur:

```
useradd adrien
```

Ou de le supprimer:

```
userdel adrien
```

Une fois créé, vous trouverez toutes les informations sur les utilisateurs du systèmes dans /etc/passwd. On peut également créer un groupe avec `groupadd` et supprimer avec `groupdel`

Utilisez `man` pour connaître comment positionner un utilisateur dans un groupe

La commande `passwd` permet de changer le mot de passe d'un utilisateur

Gestion des groupes

Il est possible de changer de propriétaire:

```
chown adrien monfichier
```

Ou de changer de groupe:

```
chgrp users monfichier
```

Les droits avancés

Trois droits spéciaux:

- `setuid` : Lorsqu'un utilisateur exécute un programme, celui-ci se lance avec les droits de cet utilisateur (exemple: passwd appartient à root, un utilisateur lamba peut le lancer en tant que root)
- `setgid` : même principe que `setuid` mais pour les groupes
- `sticky bit` : Lorsque l'on positionne le sticky bit, un exécutable restera en mémoire même lorsqu'il aura terminé son exécution, ainsi, il se lancera plus rapidement au prochain lancement. Cette pratique tend à être obsolète. Seul l'administrateur système peut positionner le sticky bit.

Gestion des processus

Loïc Guillois

Gestion des processus

PID :

Un process est un programme en cours d'exécution. Il lui est attribué un PID (process ID), un nombre qui le caractérise de manière unique.

Les commandes de gestion de processus

Informations sur les processus en cours :

`ps -u` : Affiche la liste de tout les process dont l'utilisateur est propriétaire, et des informations comme le PID,occupation mémoire, conso CPU, etc...

`top` : Liste en temps réel les process sur l'ordinateur, et ressources utilisées.

La commande `htop` n'est pas une commande disponible par défaut mais dispose d'un affichage beaucoup plus pratique

Les commandes de gestion de processus

Informations sur les processus en cours :

Crtl-C: interrompt le process en mode interactif.

`kill -9 8564` : Tue le process de PID 8564.

`renice 20 -p 8564` : Fixe à 20 la priorité du process de PID 8564.
(20 = priorité la plus faible)

Pour tuer un processus attaché à une fenêtre, on peut utiliser `xkill`

Les commandes de gestion de processus

Lancer un process en tâche de fond :

`gzip grosfichier &` : Lance la compression d'un fichier lourd en tache de fond

On peut cascader directement des commandes :

```
echo toto & date & echo toto
```

Les commandes de gestion de processus

Mettre un process en suspend :

Contrôle-Z : Suspend l'exécution d'un process lancé en mode interactif (code, éditeur, navigateur, etc...)

`jobs` : Donne la liste des process suspendus.

`fg %1`: Reprend, en interactif, l'exécution du process suspendu numéro 1 (liste de jobs).

`bg %1`: Reprend, en tâche de fond, l'exécution du process suspendu numéro 1.

Essayez de lancer la commande yes et de la mettre en suspend puis de la réactiver

Les commandes de gestion de processus

Exécution en cascade :

Si l'on veut effectuer des opérations en cascade avec la condition que la tâche précédente se soit déroulée avec succès, on utilise `&&`

```
backup.sh && send.sh && notify.sh
```

Si une erreur se produit lors de l'exécution de backup.sh alors send.sh et notify.sh ne seront pas exécutés. Si une erreur se produit lors de l'exécution de send.sh alors notify.sh ne sera pas exécuté.

Le shell

Loïc Guillois

Généralités

Le shell :

Le shell permet de tapper et d’exécuter des commandes.

Il ne se limite pas à celà et permet également de réaliser des redirections. Une redirection consiste à rediriger l'entrée ou la sortie d'une commande vers une autre commande ou un fichier.

Les jokers

Caractères spéciaux utiles :

? : remplace un caractère quelconque.

* : remplace une chaîne de caractères quelconque.

Exemples d'utilisation:

`rm rep1/*.dat` : détruit tout les fichiers du répertoire rep1 qui finissent par .dat

```
mv ../data/out0?.dat ~/poub/
```

Protection des caractères spéciaux

Empêcher l'interprétation de certaines commandes :

`echo *` permet d'afficher tous les fichiers, l'étoile étant interprétée. Mais si l'on veut réellement afficher * nous devons ajouter des quotes : `echo '*'`

`echo '` provoque un résultat inattendu, si l'on veut afficher une quote alors nous devons la protéger avec des doubles quotes : `echo "'"` ou le backslash : `echo \'`. Ce dernier ce protège lui même : `echo \\`

Les redirections

Comprendre les entrées sorties :

L’entrée par défaut est le clavier

La sortie par défaut est l’écran

C’est ce qui permet au terminal de récupérer les caractères tapés au clavier et d’afficher les résultats des programmes sur l’écran.

La sortie par défaut est numéroté 1. Il existe également une sortie d’erreur numéroté 2 qui par défaut s’affiche à l’écran. Certains terminaux permettent une colorisation spécifiques.

Les redirections

Comprendre les entrées sorties :

Le shell permet de changer les entrées / sorties des programmes. C'est ce que l'on appel les redirections.

Rediriger la sortie standard (sortie écran) dans un fichier:

`ls -1 > list` : écrit la liste des fichiers du répertoire courant dans list.

`ls -1 >> list` : écrit la liste des fichiers du répertoire courant dans list mais cette fois ci en concaténant

Les redirections

Utiliser un fichier comme entrée standard: < :

`more < data.txt` : Au lieu d'interroger l'utilisateur pour des valeurs de départ, monprog.out les lit dans le fichier data.

Les redirections

Le fichier spécial /dev/null

`yes > /dev/null` : permet d'écrire dans le fichier /dev/null (autrement dit dans le vide)

Les redirections

Le fichier spécial /dev/null

`yes > /dev/null` : permet d'écrire dans le fichier /dev/null (autrement dit dans le vide)

Les tubes (pipe)

Transmettre la sortie d'une commande comme entrée à une autre : |

Exemple:

`ps -aux | more` : Utilise more pour consulter la sortie de ps -aux

On peut chaîner à l'infini les commandes pour créer des traitements complexes.

Historique des commandes

Gagner du temps :

On peut parcourir l'historique des commandes tapées précédemment avec les flèches du haut et du bas.

La commande `history` permet de lister les dernières commandes.

Le bang permet de récupérer la dernière commande exécutée avec ses derniers paramètres.

Par exemple si vous avez tapé `echo 'hello world'`, il suffit de tapper `!echo`

Attention, le bang est dangereux. On ne peux pas visualiser la commande qui sera exécutée avant son exécution.

Expressions régulières (grep...)

Recherche de texte dans un fichier :

grep est un filtre. Il peut trouver un mot dans un fichier, par exemple :

`grep malloc *.c` : cherche la chaine de caractères malloc dans tous les fichiers dont le nom se termine par .c (*.c).

Expressions régulières (grep...)

La conception du pattern :

Caractère	**Signification**
[...]	Plage de caractères permis.
[^...]	Plage de caractères interdits.
^	Début de ligne.
.	Un caractère quelconque, y compris un espace.
*	Caractère de répétition, agit sur le caractère placé avant l'étoile. Accepte également l'absence du caractére placé devant lui.
$	Fin de ligne.
\{...\}	Répétition.
\{Nombre\}	Répétition de *Nombre* exactement.
\{Nombre,\}	Répétition de *Nombre* au minimum.
\{Nombre1 Nombre2\}	Répétition de *Nombre1* à *Nombre2*.

Expressions régulières (grep...)

Quelques exemples :

[a-z]+ cherche toutes les lignes contenant au minimum une lettre en minuscule. Le critère avec grep aurait été [a-z][a-z]*.

^[0-9]\{3\}$ cherche toutes les lignes contenant uniquement un nombre à 3 chiffres.

:[0-9]\{2,\}: cherche toutes les lignes contenant des nombres de minimum 2 chiffres avant les deux points (``**:**'').

^[0-9]\{1,5\}: cherche toutes les lignes commençant par des nombres de minimum 1 à 5 chiffres suivis par deux points (``**:**'').

(une|deux) fois cherche toutes les lignes contenant la chaîne ``une fois'' ou ``deux fois''.

^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,63}$ un email valide

Les alias

Encore gagner du temps:

Les alias sont des substitutions abrégées de commandes répétitives et/ou longues à taper dans la console.

Il est possible de définir vos alias dans deux fichiers cachés qui se trouvent dans votre dossier personnel:

- dans le fichier .bashrc
- ou dans un fichier .bash_aliases. Si ce dernier n'existe pas, créez-le.

Les alias

Créer un alias:

```
alias lscolor='ls --color=auto'
alias h='history'
```

: on créer un alias appelé lscolor

Les variables d’environnement

De la configuration simple à mettre en place:

Les variables d'environnement constituent un moyen d'influencer le comportement des logiciels sur votre système. Par exemple, la variable d'environnement « LANG » détermine la langue que les logiciels utilisent pour communiquer avec l'utilisateur.

La convention est de les mettre en majuscule. Pour les afficher on utilise le dollar

```
echo $LANG
```

On peut lister toutes les variables d’environnement avec la commande

```
printenv
```

Les variables d'environnement

Créer une variable d'environnement:

Pour créer une variable de shell, on utilise la syntaxe suivante:

```
TEST=2
```

Pour en faire une variable d'environnement, on utilise la commande export:

```
export TEST
```

On peut aussi le faire en une seule commande:

```
export TEST=3
```

La commande `unset` peut être utilisée pour supprimer une variable d'environnement

Les filtres

Les filtres sont des commandes spéciales qui permettre de modifier facilement du contenu texte pour en extraire de la donnée utile ou plus facile à lire. Les principaux sont:

- `head` : permet de lire l'entête d'un fichier
- `tail`: permet de lire la fin d'un fichier
- `sort` : permet de trier les lignes d'un fichier
- `wc` : permet de compter les mots ou les lignes d'un fichier
- `tr` : permet de convertir une chaîne de caractère en une autre de taille égale
- `split` : permet de découper un fichier
- `cut` : permet d'extraire certains champs d'un fichier
- `paste` : permet la fusion de lignes de fichiers
- `diff` : affiche les différences entre deux fichiers
- `cmp` : indique si deux fichiers sont identiques

Les scripts

Allez plus loin:

Les shell sont aussi des languages (basiques) de programmation. Les scripts sont de petits programmes contenus dans un fichier ascii exécutable, qui permettent d'automatiser de longues séries de commandes répétitives.

Les scripts

Exemple de script avec boucle et test:

```
#!/bin/sh
for f in *45; do
if [ ! -f "$f/sft25.dat" ] then
rm $f/sft.lis
rm $f/sft*.dat
fi
done
```

MERCI DE VOTRE ATTENTION

N’hésitez pas à me contacter :
loic@react-it.fr

Loïc Guillois