Cours GNU/Linux

La ligne de commande

Plan

1 - Premiers pas avec le shell

2 - Traitement de la ligne de commande

3 - Utilisation du shell

4 - Utilisation avancée du shell

1 - Premiers pas avec le shell

1 - Premiers pas avec le shell

Definitions

Présentation de l'environnement Unix

Commandes de bases

Le manuel

Le shell

C'est un langage de programmation **interprété**

C'est aussi l'interface utilisateurs des systèmes Unix lorsqu'il est utilisé de façon interactive :

- L'utilisateur passe une commande
- le shell traite la commande

Les shells

sh (1977) : le bourn shell, le shell historique unix

Shell fonctionnel mais peu souple et peu confortable.

csh (1978) : le c shell, historique aussi

Intègre des fonctions de confort tel que l'historique des commandes, alias

ksh (1983) : regroupe des fonctions du sh et du csh

C'est le shell de la norme posix (pour sa version 88)

bash (1989) : le shell le plus répandu

confort, simplicité, fonctions évoluées.

zsh : le shell ultime

autocomplétion complexe, correcteur de commande, recherche de fichier etc...

Le terminal 1/2

C'est l'interface permettant l'accès au shell.

A l'origine c'était un périphérique formé d'un écran et d'un clavier relié à un port série de l'ordinateur

C'est aujourd'hui soit :

une application graphique des système unix disposant d'une interface graphique (accessoires / terminal ou console)

une application sur votre poste de travail offrant un accès distant via le réseau à une machine Unix. Putty est l'outil standard de connexion au travers du protocole ssh

Le terminal 2/2

En application graphique ou via l'utilisation d'une connexion distante, le terminal est dit virtuel.

Il est composé d'une fenêtre affichant du texte dans laquelle l'utilisateur peut saisir des lignes de commandes :

- En entrée l'utilisateur saisit des lignes de commandes sur le terminal
- En sortie le terminal affiche le résultat de la commande

L'environnement d'exécution

Le compte utilisateur courant, le répertoire courant, ainsi qu'un grand nombre d’éléments que nous détaillons plus loin dans ce document constituent l'environnement d'exécution ou le contexte d'exécution du shell.

Unix permet d'isoler chaque processus dans un contexte d'exécution qui lui est propre.

La ligne de commande

`prompte$ commande argument1 argument2`

Exemple : `alan@laptop:~/$ ls -al /home/alan`

- `alan@laptop:~/$` : le prompte ou l'invite de commande
- `ls` : la commande
- `-al` : un argument de type “option”
- `/home/alan` : un argument de type “donnée”

L'espace « » est le séparateur des champs de la ligne de commande

Lorsqu'un argument contient un espace, celui-ci doit être protégé par des quottes simples ou doubles.

1 - Premiers pas avec le shell

Definitions

Présentation de l'environnement Unix

Commandes de bases

Le manuel

L'arborescence

Sous Unix tous les fichiers sont situés dans une arborescence unique.

Sa racine est nommé ”/”

Elle est formée par une succession de répertoires et sous répertoires

Les répertoires

Un répertoire est un fichier spécifique contenant d'autres fichiers et sous répertoires

Chaque répertoire contient au minimum deux entrées spécifiques :

- .. : spécifiant son répertoire parent
- . : spécifiant lui-même

Un répertoire ou un fichier est représenté par une entrée nommée dans le répertoire.

les fichiers

Chaque fichier ou répertoire est identifié par le chemin permettant de l'atteindre.

- Le chemin absolu : le chemin depuis la racine

  `/home/alan/fichier`

- Le chemin relatif : le chemin depuis le répertoire courant

  `fichier`

  Nous supposons ici que nous sommes dans le répertoire /home/alan

1 - Premiers pas avec le shell

Definitions

Présentation de l'environnement Unix

Commandes de bases

Le manuel

La commande ls

La commande « ls » liste le contenu du répertoire passé en argument, l'option « -al » fournie tous les détails sur les fichiers listés.

```
alan@laptop:~/test$ ls -al
drwxrwxr-x  2 alan alan 4096 janv.  3 15:07 .
drwxr-xr-x 26 alan alan 4096 janv.  3 15:07 ..
-rw-rw-r--  1 alan alan    12 janv.  3 15:07 fichier
```

Explications :

- le premier caractère indique le type de fichier : d : c'est un répertoire - : c'est un fichier
- la chaîne de caractères rwxr-xr-x indique les droits sur le fichier, ils seront présentés plus loin.
- Les deux nom suivants : alan alan indique le compte utilisateur et le groupe d'utilisateurs propriétaire du fichier
- le numérique suivant 12 indique la taille du fichier en octet, la taille de répertoire n'indique pas la taille des données contenue dans ce répertoire mais bien la taille de ce fichier spécifique.
- La date Janv. 3 15:07 indique la date de dernière modification du fichier
- le nom du fichier ou du répertoire

La commande more

La commande more permet d'afficher le contenu d'un fichier texte en l'affichant page par page.

Il est possible d'effectuer des recherches dans le fichier par l'utilisation de la touche / suivi de la chaîne de caractères recherchée. Les touches n et N permettent de rechercher l'élément suivant ou précédent correspondant.

la commande less est une autre commande permettant de faire les mêmes actions

Ces deux commandes disposent d'autres touches racourci :

g retour au début du fichier

G aller à la fin du fichier

[ctrl] u : remonte d'une page

[ctrl] d : descent d'une page

Les Commandes echo et cat

La commande echo affiche un texte passé en argument

```
alan@laptop:~/test$ echo un texte
un texte
```

La commande cat affiche le contenu du fichier

```
alan@laptop:~/test$ cat fichier
le contenu du fichier
```

Commandes de navigation

cd : pour Change Directory permet de changer le répertoire courant

cd /home/alan/test

utilisé sans argument il renvoie vers le répertoire principal de l'utilisateur courant (le home), le caractère spécifique ~ permet de nommer ce répertoire sans le connaître.

pwd : Pour Process Working Directory retourne le répertoire courant

1 - Premiers pas avec le shell

Definitions

Présentation de l'environnement Unix

Commandes de bases

Le manuel

Le manuel

La commande man permet de consulter le manuel d'une commande externe.

La commande suivante fournit le manuel de cette même commande.

```
man man
```

L'ensemble de ce cours peut etre lu dans les pages du manuel.

Les sections du manuel

Le manuel est divisé en 9 sections, chaque section traitant d'un type d'aide :

Section 1 Programmes exécutables ou commandes du shell
Section 2 Appels système (fonctions fournies par le noyau)
Section 3 Appels de bibliothèque (fonctions fournies par les bibliothèques des programmes)
Section 4 Fichiers spéciaux (situés généralement dans /dev)
Section 5 Formats des fichiers et conventions. Par exemple /etc/passwd
Section 6 Jeux
Section 7 Divers
Section 8 Commandes de gestion du système (généralement réservées au super utilisateur)
Section 9 Sous-programmes du noyau

Utilisation du manuel

Il est possible de faire une recherche dans le manuel, la commande suivante recherche la chaîne de caractère “sudo” dans toutes les pages du manuel

```
man -k sudo
```

La consultation du manuel se fait au travers d'une commande more, il est possible de faire des recherches dans cette page de la même façon

- La touche “/” du clavier permet de saisir la chaîne à rechercher.
- La touche “n” permet de chercher l'occurrence suivante
- La touche ”N” permet de revenir à l’occurrence précédente

Lecture du manuel 1/2

Le manuel se présente de la façon suivante:

```
 MORE(1)                          User Commands                          MORE(1)

NAME
     more – file perusal filter for crt viewing

SYNOPSIS
     more [-dlfpcsu] [-num] [+/pattern] [+linenum] [file ...]

DESCRIPTION
     more is a filter for paging through text one screenful at a time.  This
     version is especially primitive.  Users should realize that less(1) pro-
     vides more(1) emulation plus extensive enhancements.

OPTIONS
     Command-line options are described below.  Options are also taken from
     the environment variable MORE (make sure to precede them with a dash
     (``-'')) but command line options will override them.

     -num  This option specifies an integer which is the screen size (in
           lines).
.../...
```

Lecture du manuel 1/2

Le synopsis défini la syntaxe de la commande

```
SYNOPSIS

     more [-dlfpcsu] [-num] [+/pattern]
[+linenum] [file ...]
```

- La commande
- Les options et arguments : lorsqu'entre crochets, ils sont facultatifs. Ils sont en général expliqués par la suite dans le man.
- Le symbole | signifie un “ou” logique

TD :

manuel des commandes :

- cat
- more
- grep
- file
- type
- cp
- mv

2 - traitement de la ligne de commande

2 - traitement de la ligne de commande

Principe de interpretation par le shell

- Evaluation de la ligne de commandes
- Les caractères spéciaux
- Execution de la ligne de commandes

Les entrées sorties et redirections

- Les entrées sorties
- Les redirections
- Les enchaînement de commandes et la récursion

Principe

Après la saisie d'une ligne de commande et la touche entrer, celle-ci est traitée par le shell en 2 phases

1 - Évaluation de la ligne de commande

2 - Exécution de la ligne de commande

L'exécution peut retourner un texte qui s'affiche sur le terminal

Phase 1 : évaluation de la ligne 1/3

1 - Résolution de la commande : le shell évalue la commande afin de savoir comment la traiter. La commande “type” simule cette action.

- recherche d'alias
  - les utilisateurs peuvent définir des alias de commande (voir plus loin), le shell commence par remplacer l'alias par la commande associée.
- le shell recherche la commande dans ses commandes internes
  - S'il la trouve alors la résolution est terminée la commande interne sera utilisée
  - S'il ne la trouve pas il recherche alors la commande externe
- Le shell recherche la commandes dans les répertoires de commandes :
  - S'il la trouve il la remplace par le chemin complet du fichier exécutable à lancer
  - S'il ne la trouve pas le shell retouorne une erreur

Exemples

```
alan@laptop:~/$ type ll
ll est un alias vers « ls -alF »

alan@embeded-env:~/test$ type echo
echo est une primitive du shell

alan@laptop:~/$ type bash
bash est /bin/bash
```

Phase 1 : évaluation de la ligne 2/3

Evaluation des Arguments 1/2

Remplacer les arguments par leur valeur effective. La commande echo permet d'afficher le résultat de l'évaluation des arguments.

Valorisation des variables

Remplace les variables par leur valeur effective

```
alan@laptop:~/$ echo $HOME
/home/alan
```

Traitement des wildcards (voir plus loin)

Effectue une résolution de nom de fichier en remplaçant les caractères spéciaux

```
alan@laptop:~/$ echo /home/*
/home/alan /home/user
```

Phase 1 : évaluation de la ligne 3/3

Evaluation des Arguments 2/2

Pré-exécution de commandes

- Un argument peu être le résultat d'une autre commande

```
alan@laptop:~/$ echo $(ls -ald $HOME/Documents)
drwxr-xr-x 2 alan alan 4096 janv. 1 17:05
/home/alan/Documents
```

- Cette commande sera alors aussi évaluée puis exécutée avant le traitement de la ligne de commande complète.

Les caractères spéciaux 1/2

Les caractères spéciaux sont interprétés par le shell, ils représentent une fonction particulière dans l'interprétation de la ligne de commande.

Les caractères propres au langage

<space> <tab> <newline> : les séparateurs

$: défini une évaluation à effectuer (sera traité dans plusieurs sujets suivants : variables, substitution, exécution préalable)

> < : ces caractères permettent la gestion des entrée sorties et des redirections (ils seront traité plus loin)

Les opérateurs de contrôles de commandes, ils permettent d'enchaîner les commandes

|| & && ; ;; ( ) | |&

Caractères spéciaux 2/2

Les opérateurs sur l'évaluation de la ligne de commande :

- # : ne pas traiter la suite de la ligne (mise en commentaires)
- / : supprime le caractère spécifique du caractère suivant
- " et ' : encapsulation de texte avec un espace, ils permettent de traiter une chaîne de caractère avec un espace comme un seul argument
  - “texte“ : le texte sera évalué avant l'exécution
  - 'texte' : le texte ne sera pas évalué avant l'exécution
- ainsi nous pourrons protéger des caractères spéciaux afin qu'il ne soit pas traités par le langage

Phase2 : Exécution de la ligne

Lancement d'un processus fils du shell :

- un appel système est fait (fork) pour créer un nouveau processus

utilisation du fichier exécutable résolu

- un second appel système est fait (exec) pour charger le fichier exécutable en mémoire en lui passant les arguments évalués

Phase 2 : Exception

Les commandes internes, sont des commandes directement implantées dans le shell.

Elles ne lancent pas de processus fils mais effectuent une opération directement sur le shell

Elles permettent sa configuration et la gestion du contexte d’exécution

2 - traitement de la ligne de commande

Principe de interpretation par le shell

- Evaluation de la ligne de commandes
- Les caractères spéciaux
- Execution de la ligne de commandes

Les entrées sorties et redirections

- Les entrées sorties
- Les redirections
- Les enchaînement de commandes et la récursion

Les entrées sorties

Principe : toute commande (processus) dispose d'une entrée standard, d'arguments, d'une sortie standard et d'une sortie d'erreur.

Les Arguments :

- le texte saisie après la commande
- Les arguments sont séparés par des espaces ou tabulations

Les sorties standard et d'erreur sont en général redirigées vers le terminal de l'utilisateur, il s'agit de deux flux de données distincts

- Le flux standard : Ce que retourne la commande
- Le flux d'erreur : Les messages d'erreurs à l'exécution

L'entrée standard

- La saisie sur le terminal
- Un fichier passé en paramètre
- Un flux de sortie d'une autre commande : Le pipe

Les redirections 1/3

Les Sorties :

La sortie standard : “cmd > fichier” redirige la sortie de la commande cmd vers le fichier fichier, il est créé s'il n'existe pas

```
echo toto > sortie-std.txt
```

Le flux d'erreur : “cmd 2> fichier ” redirige les messages d'erreur vers le fichier fichier, il est créé s'il n'existe pas

```
cat truc 2> sortie-err.txt
```

La redirection simple : “>” écrase le contenu du fichier existant

La redirection double : “>>” ajoute la sortie au contenu existant du fichier

Les redirections 2/3

L'entrée standard

“cmd < fichier” : la commande lit son entrée standard dans le fichier fichier.

La commande doit être une commande qui attend un flux de données en entrée standard qui sera le contenu du fichier

```
cat < fichier
```

Les redirections 3/3

Les commandes traitant un flux de données peuvent lire ce flux sur la sortie d'une autre commande.

Le caractère spécial “|” nommé pipe ou tube permet de chaîner ces commandes sur le flux de données

Exemple :

```
echo voici un contenu | cat
```

cette commande n'a aucun autre intérêt que de présenter le pipe

TD : Les redirections de sorties

le flux de sortie standard : envois la sortie vers un fichier et le créé s'il n'existe pas

- ls -al
- echo toto > tt
- ls -al
- cat tt

le flux de sortie d'erreur : envois les messages d'erreur vers un fichier

- cat truc 2>> tt
- cat tt

Simple redirection : écrase le contenu existant

- echo toto > tt
- cat tt

Double redirection : ajoute au contenue existant

- echo toto2 >> tt
- cat tt

TD gestion de l'entrée standard

Expliquez ces 3 actions :

```
cat < ./tt
```

```
cat tt
```

```
cat tt | more
```

Enchaînements de commandes

Un séparateur de commandes : “;” permet d'exécuter les commandes l'une après l'autre :

```
ls –al ; echo voila
```

Enchaînement sur un flux de données : |

```
cat .profile | more
```

la sortie de la commande “cat .profile” est renvoyée vers l'entrée de la commande more

La récursion

l'évaluation de la commande peut impliquer une sous exécution de commande.

Les syntaxe suivante invoque une exécution préalable par le shell de la commande cmd : $(cmd) et `cmd`

```
echo nous sommes le $(date +%d/%m/%y)
```

La commande date permet d'affichier la date. La commande précédente parmet d'afficher un texte suivi du résultat de la commande date.

L’interprétation de la ligne de commande est donc faite suivant un algorithme récursif

3 - Utilisation du shell

3 - Utilisation du shell

L'environnement d'exécution

Les commandes internes

La gestion de fichiers

Présentation

Le shell comme tout processus sous unix est exécuté dans un environnement

- Le compte utilisateur et ses attributs (cf cours gestion des utilisateurs)
- Répertoire courant
- Variables d'environnement
- Les fonctions et alias définis

c'est le contexte d’exécution

Les variables

Les variables

Une variable est un mot qui définit une valeur abstraite.

Le mot commence par un caractère alphabétique. D'usage, on les définit en minuscules. Les variables en majuscule sont celles utilisées par le système.

Lorsque dans le shell on travaille sur une variable, on traite la valeur qu'elle contient.

Gestion des variables 1/3

Pour définir la variable : **var** et lui affecter la valeur : **ma-valeur :**

```
var=ma-valeur
```

Pour utiliser une variable et donc traiter sa valeur nous la préfixons par un **$** :

```
$var
```

Lorsque le nom de la variable contient des cratères spéciaux on place le nom de variable entre accolades

```
${v-ar}
```

Gestion des variables 2/3

Une variable non définie est utilisable et renvoie une valeur nulle : une chaîne de caractère vide

```
echo $toto
```

Pour supprimer la variable **var :**

```
unset var
```

Pour bloquer une variable en lecture seule

```
readonly var
```

Celle-ci ne sera ni modifiable ni suppressible

Gestion des variables 3/3

Une variable n'est utilisable que dans le processus courant (votre shell)

Afin de mettre une variable à disposition de tous les futurs processus fils

```
export var
```

Les variables internes

Ces variables sont réservées, accessibles en lecture et valorisées par le shell

- $0 : le nom du script tel qu'il est appelé
- $1 .. $9 .. $N : les arguments de la ligne de commande
- $# : le nombre d'arguments
- $* et $@ : tous les arguments (“$@” représente “$1” “$2” ...)
- $? : le code retour de la dernière commande exécutée
- $! : le numéro de processus de la dernière commande lancée en arrière plan
- $$: le processus ID du shell courant

Les Alias

Les alias permettent le remplacement d'un alias de commande par une commande complexe lors de l'évaluation de la ligne.
Exemple : 'll' se traduit par la commande 'ls -al'

Syntaxe : alias cmd='commande complexe'

Passée sans argument la commande “alias” retourne les alias déjà définis (à l'initialisation du shell)

La commande “unalias” permet de supprimer un alias

```
unalias ll
```

Configuration

Lors de son initialisation, le shell *source* les fichiers de configurations associés

Sourcer c'est exécuter toutes les commandes définies dans ces fichiers dans le shell courant

- Au système : /etc/profile /etc/bashrc etc...
- Au compte utilisateur ~/.profile ~/.bashrc etc...

Initialisation du contexte d'exécution, exemple :

- la variable PS1 qui définit la forme du prompte
- Les couleurs associées aux types de fichiers pour la commande ls
- Définition d'alias

Gestion de l'environnement

La commande “set” permet d'afficher toutes les variables et fonctions présentes dans l’environnement

Certaines sont exportées pour être visibles par les commandes externes.

La commande “env” ou “printenv” permet d'afficher les variables exportées

La variable PATH

La variable PATH a un rôle important dans le fonctionnement du shell

Elle définit lors de la résolution de commande :

- Les répertoires contenant les commandes
- L'ordre de recherche dans ses répertoires

Elle permet donc d'appeler les binaires sans fournir le chemin complet de ceux-ci

Attention : il est possible en changeant cette variable de vous faire exécuter de mauvaises commandes.

Autres variables internes au shell

LOGNAME : définit le nom du compte utilisateur au login

USER : définit le nom du compte utilisateur courant

UID : définit l'identifiant numérique du compte utilisateur

HOME : définit le répertoire principal de l’utilisateur

LANG : définit le langage utilisé par le shell

PID : définit l'identifiant numérique du processus en cours d'exécution

PPID : définit l'identifiant numérique du processus père du processus courant

PWD : définit le répertoire courant du shell

TERM : définit le type de terminal actuellement utilisé

COLUMNS / LINES : définit le nombre de colonnes/lignes du terminal

3 - Utilisation du shell

L'environnement d'exécution

Les commandes internes

La gestion de fichiers

Commandes internes 1/4

Les commandes internes ne disposent pas de page de manuel, elle sont intégrées dans le shell et seront donc documentées dans le manuel du shell. Une aide rapide sur la commande cmd est visible via la commande help cmd :

help

help help

On peut en distinguer 3 classes :

- Commandes de gestion de l'environnement du shell
- Commande de gestion du comportement du shell
- Commandes de contrôle du shell

Commandes internes 2/4

Gestion de l'environnement :

pwd / dirs / cd / pushd / popd : gestion du répertoire courant et de la pile de répertoires

alias / unalias : gestion des alias

declare / typeset / set / unset / local / export / readonly : gestion des variables (cf chapitre associé)

source / “. “ file : lance les lignes commande du fichier file dans le shell courant

umask : gestion du umask (cf gestion des droits)

getops / shift : gestion de la ligne de commande du shell courant (ou du script)

Commandes internes 3/4

Gestion du comportement du shell

builtin / command cmd : interprete la commande cmd comme interne ou comme une commande externe

eval $cmd 'cmd' : effectue une pré-évaluation avant le traitement standard de la commande la partie entre simple quotte ne sera pas pré-évaluée

exec cmd : lance la commande cmd à la place du shell courant (sans créer de processus fils) le shell courant une fois remplacé n'existe plus.

exit / logout 42 : termine le shell avec le code retour 42

trap cmd 1 2 3 : Piège les signaux kill 1, 2 et 3 et les traite en passant la commande cmd

Commandes internes 4/4

Contrôle du shell :

fg / bg / jobs / wait / kill : gestion des processus lancés en arrière plan

for / do / done / select / while / until / break / continue : structure de contrôle du langage

function / return : gestion des fonctions

if / case / test : gestion des tests et conditions

echo / printf : génération de sortie texte

read / read array : lecture de l'entrée du shell

3 - Utilisation du shell

L'environnement d'exécution

Les commandes internes

La gestion de fichiers

Les fichiers 1/3

Sous Unix tout est fichier (ou presque).

La commande “ls”

- syntaxe : ls [options] [files]

Options courantes : “-altr” avec les options a pour all, l pour long, t pour time, r pour reverse

Arguments : une liste de fichiers

- Si le fichier est un répertoire : liste le contenu du répertoire
- Si le fichier est un simple fichier : liste les attributs standard sur le fichier

Les fichiers 2/3

Sortie le la commande ls -al

```
alan@laptop:~/test$ ls -al
total 12
drwxrwxr-x  2 alan alan 4096 janv.  3 15:07 .
drwxr-xr-x 26 alan alan 4096 janv.  3 15:11 ..
-rw-rw-r--  1 alan alan    5 janv.  4 13:35 fichier
```

Les fichiers 3/3

Attributs de fichier en sortie de commande “ls -l”

Le Type de fichier:

- pour un simple fichier

d pour un répertoire

l pour un lien symbolique

Les Droits sur le fichier : Owner, group-owner, others

- Sur fichier : rwx : Read Write Execute
- Sur répertoire : r lister le contenu, x traverser, w créer/supprimer une entrée dans le répertoire

Le propriétaire le groupe

la taille en octet (changer les options pour avoir un autre format)

la date de dernière modification

le nom du fichier

Les formats de fichiers 1/2

Les format de fichiers ne sont pas déterminés par une extension au nom de fichier.

La commande file détermine par une séquence d'évaluation du fichier son format.

Les types de fichiers 2/2

Les fichiers texte : Se sont des simples fichiers ascii

Les fichiers compressés / archives : fichiers sous format standard regroupant un ou plusieurs fichiers

Les fichiers exécutables : les commandes externes

Les fichiers de données : fichiers sous tout autre format

Appels de fichiers

Nous avons vu les chemins relatifs et absolus. Il est aussi possible d'omettre une partie d'un chemin ou d'un nom de fichier en le remplaçant par un caractère générique “wildcard”

les wildcards ressemblent aux expressions régulières mais sont largement simplifiées

voici les principaux :

* toute chaîne de caractère

? un caractère

[arf] un caractère parmi a r et f

[a-f] un caractère compris entre a et f

Exemples

ls ~/*.txt

ls ~/*/*.txt

lecture de fichiers

file : définit le type de fichier

head : retourne le début d'un fichier

tail : retourne la fin d'un fichier

cat : retourne le contenu des fichiers passés en arguments

more/less : retourne le contenu du fichier formaté page par page

nl : retourne le fichier en numérotant les lignes

wc : compte le nombre de lignes ou de mots

Gestion des répertoires

La commande “mkdir” permet de créer un répertoire. L'option “-p” permet de créer toute la sous arborescence

```
mkdir –p ~/test/rep1/rep2
```

La commande “rmdir” permet de supprimer un répertoire vide. Un répertoire non vide sera supprimé avec la commande “rm”

```
rmdir ~/test/rep1/rep2
```

Manipulation de fichiers

La commande “cp” permet de copier des fichiers. Elle prends en argument un fichier ou une liste de fichiers et la destination de la copie

```
cp fichier ~/test/rep1/copiedefichier
```

```
cp fichier ~/test/rep1/copiedefichier ~
```

la destination peut être un nom de fichier (pour un seul fichier source) ou un répertoire de destination

La commande “mv” permet de renommer ou déplacer un fichier, elle fonctionne de la même façon que la commande cp, mais ne conserve pas le fichier original.

Suppression de fichiers

La commande “rm” permet de supprimer un fichier. L'option “-f” permet de forcer l'action et l'option “-r” permet d'effectuer l'opération récursivement sur la sous arborescence.

```
rm –rf ~/test/rep1
```

Attention cette commande est destructive il n'y a pas de retour arrière possible

Suppression de fichiers

La commande “rm” permet de supprimer un fichier. L'option “-f” permet de forcer l'action et l'option “-r” permet d'effectuer l'opération récursivement sur la sous arborescence.

```
rm –rf ~/test/rep1
```

Attention cette commande est destructive il n'y a pas de retour arrière possible

Compression de fichier

Les commandes : compress, gzip, bzip2 permettent de compresser un fichier, il est alors remplacé par le fichier compressé.

Exemple : `bzip2 ./fichier`

Les commandes : uncompress, gunzip, bunzip2 permettent de décompresser un fichier, il est alors remplacé par le fichier décompressé.

Exemple : `gunzip ./fichier.gz`

La différence entre ces commandes est le format de compression utilisé.

Attention ces commandes utilisent l'extension de fichier .Z .gz .bz2

Archive de fichiers

La commande "tar" permet d'archiver toute une sous arborescence dans un fichier unique

- tar cvf une/archive.tar des fichiers à mettre dedans
  - créé une/archive.tar contenant les fichiers et sous arborescences : ./des, ./fichiers, ./a, ./mettre et ./dedans
- tar xvf une/archive.tar
  - extrait une archive.tar dans le répertoire courant
- tar xvf une/archive.tar un-fichier
  - extrait un-fichier d'une/archive.tar
  - (note : les options c, x, v et f signifient Create eXtract Verbose File)
- tar xvf - : permet d'extraire une archive lue sur entrée standard.
  - Exemple : bzcat une-archive.tar.bz2 | tar xvf -
- tar zxvf une/archive.tgz
  - décompresse au format gzip puis extrait l'archive
- tar jcvf une/archive.tar.bz2 liste fichiers
  - créé l'archive compressée au format bz2

4 - Utilisation avancée du shell

Gestion des droits sur les fichiers

Commandes filtres et pipeline

Présentation 1/2

Chaque fichier et répertoir définis les droits qui lui sont associées pour le propriétaire (owner), les membre du groupe propriétaire (group) et tout les autre (others)

Signification des droits :

- Sur fichier : rwx : Read Write Execute
- Sur répertoire : r lister le contenu - x traverser le répertoire - w créer un fichier dedans

Doits avancé :

- Le set uid bit : Binaire : autorise la prise d'identité du owner (si le binnaire fait l'appel system correspondant)
- set gid bit : Répertoire : tout fichier créer dans ce répertoire hérite du group owner du répertoire
- sticky bit : Répertoire : supression limité au créateur

Présentation 2/2

Codage des droits :

Les droits sont un ensemble de flag sur le fichier aillant une valeur à 0 ou 1

Il sont en général représentés en octal, avec 4 valeurs de 0 à 7

Pour chacun des User, Group et Other : Un nombre octal correspondant à la suite des 3 flags :

Read oui : 1 , Write non : 0 , Execute oui : 1

soit en binaire 0x101

donc en Octal : 1*4+0*2+1*1 = 5

Exemple de codage complet : 754 rwxr--xr--

Pour les droits avancés les set uid, setgid et stiky bit sont codé sur un premier octal optionel :

0 si aucun ne sont positionné (on ommet alors de le préciser)

Puis de la même façon 100 devien 5, 100 deviens 4 001 deviens 1

TD

Representez en octal les droits suivants

- rw-rw----
- rwsrwxr-x (c'est pas joli joli)
- rw-rw-r-t
- rwxr-sr-x

Representez litéralement les droits suivants

- 755
- 644
- 2775
- 4775

Commandes de gestion des droits

chown : change le owner **utilisable par root uniquement**

chgrp : change le groupe owner

chmod : modifie les droits sur le fichier/répertoir

Notation en mode :

- 777 !! non !!
- 755 - 750 std

Notation litéral

- u+rwxs
- g+xs
- o+t

TD

Lisez le manuel des commandes :

- chown
- chmod
- chgrp

testez les !

Groupes

Les groupes sont définis sur les utilisateur à la connexion

Le GID du compte est l'ID du groupe primaire

- Les fichiers créer ont pour groupe le gid du compte
- Sauf en cas d'usage de set gid bit sur le répertoire

Testez :

- Le le set gid fonctionne-t-il si le compte ne fait pas parti du group
- Comment vous testez?

Le umask

Cette commande definie la valeur par défaut du mode pour tout nouveau fichier ou répertoire

umask : retourne le umask courant

umask 022 : spécifie le umask à la valeure 022

C'est le complément octal du mode : 022 droits par défaut : 755

expliquez les umask : 0027 / 027 / 022 / 007

Attention : le droits d'execution n'est jamais mis par défaut à la création de fichier il doit être ajouté explictement (chmod u+x).

C'est aussi une variable utilisée dans les fichier de configuration de service (exemple ftp) pour spécifier les droits sur les fichiérs créé par ce service.

TD

Comment effectuer un transfert de fichiers entre 2 utilisateurs ?

Ana a le fichier photo.jpg dans son répertoire /home/ana, bob doit faire une retouche sur la photo, la sauvegarder sous photo-ret.jpg et transmetre le fichier à ana.

Definir l'ensemble des actions de bob et ana sur les droits des 2 fichiers

4 - Utilisation avancée du shell

4 - Utilisation avancée du shell

Gestion des droits sur les fichiers

Commandes filtres et pipeline

filtres

Commandes traitant les flux de données en entrée et les renvoyant sur la sortie

- more / less : affiche page par page
- wc : compte les mots / les lignes
- sort : effectue un tri

Rappel : Pipelines / tubes / le pipe / |

- Test :
  - ls -l > resultat.txt
  - wc -l < resultat.txt
  - Quels sont les défauts de cette méthodes ?
- Le pipe
  - ls -l | wc -l

Rappel : Les entrées sorties

Principe : toute commande (processus) dispose d'une entrée standard, d'arguments, d'une sortie standard et d'une sortie d'erreur.

Arguments :

- le texte saisie aprés la commande
- Les arguments sont séparrer par des espaces ou tabulations (IFS)

Sorties standard et d'erreur

- En général redirigé vers le terminal de l'utilisateur
- Std : Ce que retourne la commande
- Err : Les messages d'erreur à l'execution

Entrée standard

- La saisie sur le terminal
- Un fichier passé en parametttre
- Un flux de sortie d'une autre commande : Le pipe

Enchainements de commandes sur un flux de données

Exemple sur le flux standard :

```
cat tt | more
```

Exemple sur le flux d'erreur :

```
cat toto 2>&1 | grep toto
```

Les filtres

basename / dirname

Extraction des noms et chemin sépare dans une chaine de caratère nommant un fichier

Entrée : argument : nom complet de fichier

Sortie : sortie standard

- Le répertoire ou il se trouve : dirname
- Le nom du fichier : basename

grep

Extraction des lignes d'un fichier texte validant ou invalidant des critères de recherche

Entrée :

- Les critères de recherches en arguments
- Un fichier en dernier argument ou le flux d'entrée standard

Sortie :

- Les lignes correspondant à la recherche codée par les critères

Critère de recherche standard : contien la chaine de caratère en premier argument

- Exemple : ls -al | grep toto

Options :

- -i ne tiens pas compte des majuscule et minuscule
- -v , inverse la sélection, retourne les lignes qui ne contienent pas la chaine
- -c comptes les occurrence au lieu de les afficher (équivaut à “| wc -l”)

Expressions régulières

Une expression régulière est une chaine de caractère définissant un ensemble de chaine de caratère : c'est un critère de recherche trés évolué

Basé sur des caratères spéciaux

- ^ : début de chaine
- $: fin de chaine
- [a-z|A-Z] : un caratère alphabétique (| est donc le ou logique)
- [0-9] : un caratère numérique
- Quantificateurs :
  - + au moins une fois
  - * un nombre quelquonque de fois
  - ? 0 ou 1 fois

egrep / fgrep

fgrep équivalent à grep -F le critère est une simple chaine de caractère (fast grep)

egrep équivalent à grep -E (et souvent à grep) : Le critère de recherche est une expression régulière.

Exemples :

ls -al | egrep “^d”

ls | egrep “.txt$”

ls | egrep -v [0-9]

ls /etc/rc3.d/ | egrep ^S[0-9]

cut

Outil de découpage des lignes

Entrée :

- Arguments : option de découpage
- Flux de données ou fichier nommé en dernier argument

Sortie standard : résultat du découpage

Exemple :

- cut -c1-5 /etc/passwd
- cut -d: -f3,6 /etc/passwd

Décompte

wc : word count

Options :

- -l les lignes
- -m les caratère
- -c les octet
- -w les mots

Exemple :

- wc -l /etc/group

nl : number of line

nl /etc/group

sort : tri

Par défautn cette commande trie par :

ordre croissant

ordre alfanumérique

Options :

-d : trie suivant l'ordre du dictionaire

-n : trie par ordre numérique

-f : ne tiens pas compte des majuscule dans le tri

-r : reverse l'odre par défaut : ordre décroissant

-k n : tri suivant la colone n

-t: : le séparateur de champs est “:”

Exemple :

sort -t: -k 4 -n -r /etc/passwd

Uniq : supprime les doublons

Cette commande supprime les répétition de lignes consécutives

elle est en général utilisé en filtrage de sortie d'un sort

Exemple

- cut -d: -f4 /etc/passwd | sort -n | uniq
- cut -d: -f4 /etc/passwd | sort -n | uniq -c

Search Replace : tr / sed

tr : translate traduit les caratères

cat passwd | tr “olzeasbtg” “012345679” | tr “OLZEASBTG” “012345879”

Options : -s supprime les répétitions

- cat /etc/fstab | egrep -v "^#" | tr -s " " ";"

sed : stream editor : éditeur de flux

Outil ultra évolué

Exemple : sed 's/search-regex/replace/g'

jointure de fichiers
cat / paste / join

cat fica

toto:1
totu:2
tutu:3
titi:4
tata:5

cat ficb

t0:to:1
t0:tu:2
t0:tu:3
t0:ta:5

cat fica ficb

paste : copie ligne a ligne 2 fichiers : paste -d";" fica ficb

toto:1;t0:to:1
totu:2;t0:tu:2
tutu:3;t0:tu:3
titi:4;t0:ta:5
tata:5;

join : effectue la jointure sur un champ commun : join -t: -1 2 -2 3 fica ficb

1:toto:t0:to
2:totu:t0:tu
3:tutu:t0:tu
5:tata:t0:ta

TD : Découpe de fichiers : commande split

Découpez un fichier en morceau de 5 octet et reconstruisez le.