Unix - Exploitation

UX10 – *B0039AER20141030*

Xerox Formation
Immeuble "Le Cardinet"
8, rue Bernard Buffet – 75017 PARIS
Tel : 01 55 30 62 00 – Fax : 01 55 30 62 30
http://xerox-formation.fr

I. Introduction aux systèmes UNIX

I.1 Historique des systèmes UNIX

1965 Mise au point de "**Multics**" (**Mult**iplexed **I**nformation and **C**omputing **S**ystem) par **Ken Thompson** (laboratoires Bell AT&T), un système interactif simple destiné à faire tourner le jeu "Space Travel" (simulateur de système solaire).

1969 Les laboratoires Bell **AT&T** abandonnent le projet "**Multics**".
Ken Thompson et **Dennis Ritchie** réécrivent une version simplifiée de Multics, baptisée plus tard **UNICS** par **Brian Kernighan**. Cette version est destinée à faire tourner leur jeu sur une petite machine disposant de très peu de mémoire pour les programmes utilisateur (DEC PDP-7).
Le 1er Janvier 1970 est considéré comme étant la date de **naissance du système UNIX**.

1973 Mise au point du **langage C** par **Dennis Ritchie** et **Brian Kernighan**.
Réécriture du noyau Unix en langage C, puis de tous les autres outils utilisés sous Unix (Version **Unix Time-Sharing System V4)**.
Distribution des sources Unix dans les universités, à des fins éducatives, notamment l'université de **Berkeley**.

1977 Deux branches de développement de sources Unix, voient le jour :
- **BSD** (**B**erkeley **S**oftware **D**istribution) dont la première version sera basée sur **Unix Time-Sharing System V6** et développée par **Bill Joy**.
- **AT&T** qui allait donner naissance à **Unix System V**.
  Les sources Unix d'AT&T sont par ailleurs mises à disposition d'un certain nombre d'entreprises, notamment les **constructeurs informatiques**.

1979 **AT&T** distribue gratuitement son **Unix Time-Sharing System V7** (version majeure à l'origine de la plupart des versions suivantes) :
- Support du "swapping".
- Support des fichiers de grande taille.
- Intégration au système du **compilateur C** et du **Bourne Shell**.

1979 La version concurrente de **BSD** intègre :
- La **mémoire virtuelle** et la pagination.
- Les protocoles **TCP/IP**.

1979 à 1990 Développement de **systèmes "Unix-like"** par les contructeurs informatiques :
- **AIX** d'**IBM** (à partir de 1990) basé sur le **System V**.
- **SOLARIS** de **SUN** (à partir de 1982) basé sur le **System V** et **BSD**.
- **HP-UX** de **Hewlet Packard** (à partir de 1986) basé sur **BSD.**
- **SCO** de **Santa Cruz Operations** et **Hewlet Packard** (à partir de 1979) basé sur le **System V**
- **ULTRIX** de **DEC**.

1981 Des étudiants de Berkeley (dont Bill Joy) créé la société **SUN** ainsi que leur propre version d'UNIX (**SunOS 1**.O basé sur BSD 4.1).

1983 **AT&T** obtient l'autorisation de commercialiser son Unix (un décret de 1956 empêchait l'enreprise de commercialiser autre chose que des équipements téléphoniques), ce qui donne naissance à la version **Unix System V**.
AT&T propose le **standard SVID** (**S**ystem **V** **I**nterface **D**efinition).

1984 L'**IEEE** (**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers) publie une série de standards donnant naissance à **POSIX** (ou IEEEP1003) qui devient synonyme de "système ouvert".

1985 **Andrew Tannenbaum** développe un système d'exploitation minimal, baptisé **Minix**.

1987 Un consortium de constructeurs (Sun, Hp, Dec, Ibm, At&t,…) créé l'association **X/OPEN** et propose le standard **XPG** (**X**/**O**pen **P**ortability **G**uide).

1988 Création d'**OSF** (**O**pen **S**oftware **F**undation) autour d'IBM et développement de l'Unix **OSF/1**.

1989 **AT&T** propose ce qui sera son ultime version, l'**Unix System V Release 4** (**SVR4**). Cette version intègre les spécificités des Unix BSD. Elle sera adoptée par de nombreux constructeurs.

1991 **Linus Torvalds** conçoit un système d'exploitation, basé sur **Minix**, et capable de fonctionner sur les architectures de type "386". Il sera baptisé "**LINUX**"

1992 La compagnie **USL** (**U**nix **S**ystem **L**aboratories) dont **AT&T** est actionnaire majoritaire, propose **Unix System V Release 4.2**.
X/OPEN publie **XPG4**.

1993 **Berkeley** propose ce qui sera son ultime version, la **BSD 4.4**.
Novell achète **USL** et publie la version **SVR4.2MP**.
Novell cède les droits de la "marque" UNIX à **X/Open**, qui proposera un an plus tard "**The single Unix Specification**".

1995 Début des unix **OpenServer 5.0** (**SCO**) et **Digital Unix**.
X/Open propose la norme **UNIX 95**, qui regroupe les principaux standards existant.

1996 **OSF** et **X/Open** fusionnent pour former l'**Open Group**.

1997 L'**Open Group** publie la **version 2** de "**The single Unix Specification**", intégrant notamment l'achitecture des processeurs **64 bits**.
Distribution des Unix **AIX 6.4**, **HP-UX 11** et **IRIX 6.4**.

1998 L'**Open Group** propose **UNIX 98**.

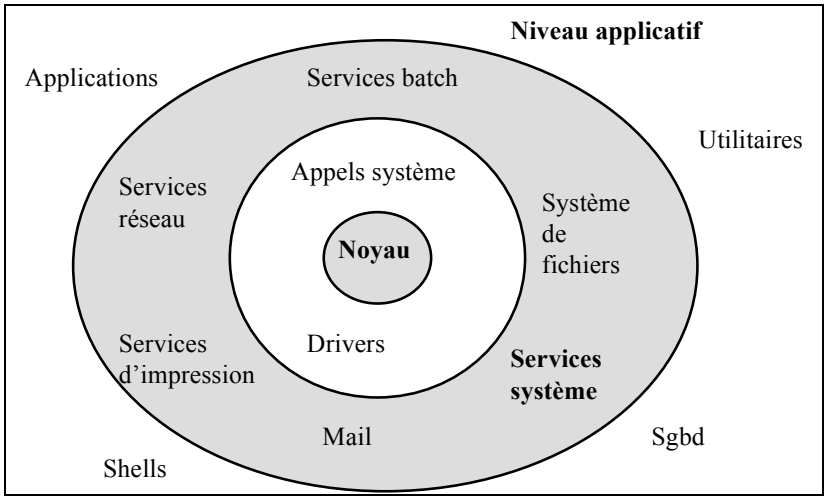
1999 **UNIX a 30 ans !!!**
Distribution du noyau de **LINUX 2.2**.
L'**Open Group** et **IEEE** s'unissent pour mettre à jour la "**Single Unix Specification**" et la norme **POSIX**.

2001 Distribution du noyau de **LINUX 2.4**.
Publication de la **version 3** de la "**Single Unix Specification"**, issue de l'union de **IEEE**, **Open Group** et des **constructeurs**.

2003 La **version 3** de la "**Single Unix Specification"** devient le **standard internationnal** (ISO/IEC 9945-2003).
Distribution du noyau de **LINUX 2.6** et de **Solaris 9.0E**.

To be continued...

I.2 Caractéristiques d'un système UNIX



UNIX est un système **multi-utilisateurs** et **multi-tâches**, présentant les principales caractéristiques suivantes.

- Un **noyau** assurant la gestion des ressources physiques (processeur, mémoire, espace disque…) et des ressources logiques (processus, arborescence, sécurité…) du serveur.
  Il dispose d'un ensemble de fonctions permettant à un programme écrit en langage C d'invoquer le noyau; ce sont les "**appels système**" (ou "primitives").

- Des **pilotes** (ou "drivers") chargés de la gestion des périphériques.

- Un **système de fichiers** hiérarchisé, permettant de localiser un fichier quel que soit son type et son emplacement physique.

- Un environnement complet de développement avec, notamment, des **éditeurs** puissants et des **compilateurs** C et C++.

- Plusieurs interpréteurs de commandes : les **shells**.
  Ils permettent à l'utilisateur d'exécuter des commandes à partir d'une ligne de commandes ("**prompt**"), mais également d'écrire des programmes : les **scripts shells**.

II. Connexion au système et session UNIX

II.1 Connexion des terminaux au serveur UNIX

Les deux modes de connexion suivants sont **obsolètes** :

- La liaison asynchrone directe. Le terminal est physiquement lié au serveur (liaison filaire en série). Le terminal est de type texte, identifié par un fichier spécial **/dev/tty*n***.
- La liaison par serveur. Le terminal est connecté au serveur Unix via un serveur de terminaux.

Aujourd'hui, le terminal est un micro-ordinateur qui émule un terminal. Il est connecté au serveur UNIX à travers un réseau TCP/IP. On peut rencontrer 2 cas de figure.

- Un poste client établit la connexion au serveur Unix, via un **protocole ssh** (par exemple). Il s'agit dans ce cas d'un pseudo-terminal, identifié par un fichier spécial **/dev/pts/*n***.
- Utilisation d'un **terminal X**, donnant accès à **l'environnement graphique** d'Unix.

II.2 Ouverture d'une session

Quel que soit le mode de connexion au serveur Unix, l'utilisateur est invité à saisir le nom de son compte (**logname**) ainsi que le mot de passe qui lui est associé.
Une partie de l'écran de "**login**" est stockée dans le fichier **/etc/issue**, et peut donc être personnalisé par l'administrateur.

```
Welcome to Ubuntu 13.04 (GNU/Linux 3.8.0-26-generic i686)

 * Documentation:  https://help.ubuntu.com/

43 packages can be updated.
30 updates are security updates.

Last login: Mon Oct 27 11:24:43 2014 from 10.236.226.179
```

Chronologie des principaux évènements liés à l'ouverture d'une session

- Informations concernant le dernier "login" et nom (ou adresse IP) du terminal.
- Affichage du "***Message Of The Day***" contenu dans le fichier **/etc/motd**.
- Invocation du **shell** associé au nom de connexion.
- Exécution des fichiers d'initialisation chargés de la mise en œuvre des **variables d'environnement**.
- Affichage du "***prompt***" (ligne ou invite de commande) indiquant que le shell est prêt à exécuter une commande. Le prompt peut être personnalisé par l'utilisateur.
  Son aspect est stocké dans la variable **PS1**.

II.3 Les principaux shells

Le shell est un processus permettant d'interpréter et d'exécuter toute commande saisie au clavier. Il constitue un interface entre le noyau et l'utilisateur. Celui-ci peut choisir parmi les shells suivants.

- Le ***Bourne-Shell*** (**AT&T**-System V) développé par **Steve Bourne**.
  Il est commun à tous les Unix. Il est invoqué par la commande **sh**.
- Le ***C-Shell*** (**BSD**) développé par **Bill Joy**.
  Mise en œuvre de fonctionnalités importantes comme la **gestion de l'historique** des commandes, le **rappel des commandes**, l'utilisation des **alias**… Il est invoqué par la commande **csh**.
- Le ***Korn-shell*** (**AT&T**-System V) développé par **David Korn**.
  Il met en œuvre les extensions apportées par le C-Shell tout en assurant une compatibilité ascendante avec le Bourne-shell. Il est invoqué par la commande **ksh**.
- Le ***Bourne-again-shell*** (**Linux**) développé par **Brian Fox** et **Chet Ramey**.
  Il est à la norme **Posix**. Il est invoqué par la commande **bash**.

II.4 Historique et rappel des commandes Unix

Les commandes passées par l'utilisateur sont conservées dans un fichier, afin d'être rappelées, éditiées et réexécutées. Trois **variables d'environnement** sont impliquées dans la gestion de l'historique.

HISTFILE	Nom du fichier historique (par défaut : **$HOME/*idshell*_history**).
HISTSIZE	Nombre maximum de commandes stockées dans le fichier.
HISTFILESIZE	Taille maximum du fichier (exprimée en nombre de blocs).

Bien qu'aujourd'hui, la majorité des Unix/Linux permettent le rappel et l'édition des commandes via les touches de déplacement de curseur, certains shells (**ksh**) nécessitent encore l'activation de ces fonctionnalités.

- **set –o vi** pour utiliser certaines commandes de l'éditeur **vi** (bonne connaissance de vi, indispensable).

i	Mode insertion
j	Commande suivante
k	Commande précédente
l	Caractère suivant
h	Caractère précédent
I	Début de ligne
A	Fin de ligne

- **set –o emacs** pour utiliser certaines commandes de l'éditeur **emacs** (plus simple).

^n	Commande suivante
^p	Commande précédente
^f	Caractère suivant
^b	Caractère précédent
^a	Début de ligne
^e	Fin de ligne

II.5 La documentation Unix : la commande man

La commande **man** permet d'afficher l'aide "en ligne" d'une commande.

```
man nom_commande
```

```
$ man id
ID(1L)                    Manuel de l'utilisateur Linux                    ID(1L)

NOM
       id - Afficher les UIDs et GIDs effectifs et réels.
SYNOPSIS
       id  [-gnruG]  [--group]  [--name] [--real] [--user] [--groups] [--help]
       [--version] [utilisateur]
DESCRIPTION
       Cette page de manuel documente la version GNU de id.
       id affiche des informations concernant  l'utilisateur  indiqué,  ou  le
       processus appelant si aucun utilisateur n'est mentionné.
       Par  défaut, il affiche l'U-ID réel, le G-ID réel, l'U-ID effectif s'il
       diffère de l'U-ID réel, le G-ID effectif s'il diffère du G-ID réel,  et
       les  G-IDs  des  groupes  supplémentaires.  Chaque  valeur est affichée
       précédée d'un libellé l'identifiant, et suivie  entre  parenthèses  des
       noms de groupe ou d'utilisateur.
       Les options permettent à id de n'afficher qu'une partie de ces informa-
       tions.
   OPTIONS
       -g, --group
              Afficher uniquement le Group-ID.
       -G, --groups
              Afficher uniquement les groupes supplémentaires.
       --help Afficher un message d'aide sur la sortie standard et se terminer
              normalement.
       -n, --name
              Afficher  les  noms  de  groupe  ou  d'utilisateur à la place du
              numéro d'ID.  Nécessite -u, -g, ou -G.
       -r, --real
              Afficher les U-ID et  G-ID  réels  plutot  que  ceux  effectifs.
              Nécessite -u, -g, ou -G.
       -u, --user
              N'afficher que l'User-ID.
       --version
              Afficher  un numéro de version sur la sortie standard et se ter-
              miner normalement.
```

Commandes de défilement de l'aide en ligne

Barre espace	Page suivante
B	Page précédente
Entrée	Ligne suivante
Flèche bas	Ligne suivante
Flèche haut	Ligne précédente
g	**Début de l'aide**
G	**Fin de l'aide**
/	Recherche de chaîne

II.6 Les variables d'environnement

La commande **env** affiche les **variables d'environnement** (ou globales).

```
$ env
TERM=xterm
SHELL=/bin/bash
SSH_TTY=/dev/pts/0
USER=alain
MAIL=/var/mail/alain
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/game
s:/usr/local/games
PWD=/home/alain
LANG=fr_FR.UTF-8
HOME=/home/alain
LOGNAME=alain
_=/usr/bin/env
```

HOSTNAME	:	Nom du serveur
SHELL	:	Shell de connexion
TERM	:	Type de terminal. Sa valeur sert de clé d'accès à la base de données des définitions des terminaux
USER	:	Utilisateur en cours
LOGNAME	:	Nom de connexion
MAIL	:	Nom de la boite aux lettres
HOME	:	Répertoire de connexion
PWD	:	Répertoire en cours
PATH	:	Ensemble de chemins d'accès utilisés pour localiser les exécutables
_	:	Dernière commande exécutée

II.7 Fermeture d'une session de travail

Un **exit** exécuté dans le processus du shell de connexion, ferme la session de travail.

III. Les commandes Unix

III.1 Généralités

Il existe deux types de commandes Unix :

- La **commande externe** au shell, constituée d'un fichier référencé dans un répertoire et dont l'exécution génère, un processus propre à la commande.
- La **commande interne** au shell, dont l'exécution ne génère pas de processus.

Les règles de syntaxe sont indépendantes du type de la commande.

III.1.1 Syntaxe des commandes Unix

```
[chemin/]commande [-options] [arguments…]
```

- **Nom de la commande**
  Il doit être le premier mot de la ligne de commande. Le chemin d'accès n'est obligatoire que si le fichier correspondant à la commande ne se trouve pas dans l'un des chemins prévus par la variable **PATH**.

- **Option(s)**
  Une option permet d'adapter ou de modifier le comportement de la commande. Elle doit être précédée d'un tiret. Les options "à une lettre" peuvent être regroupées derrière un tiret.

- **Arguments**
  Consiste généralement à nommer ou à identifier un élément (fichier, processus…).
  Les arguments doivent être saisis après les options.

```
ls -a -l exercices
ls -al exercices
ls -la exercices
cat -n /etc/passwd
```

III.2 Commandes d'informations générales

III.2.1 who

Affichage d'**informations** sur les **utilisateurs connectés** :

- Nom de connexion (équivalent au résultat de la commande **logname**).
- Voie de connexion (équivalent au résultat de la commande **tty**).
- La date et heure de connexion.
- L'adresse IP (éventuellement).

Options

-H	:	Affiche la ligne d'entête de colonnes.
-m	:	Limite l'affichage aux informations concernant l'utilisateur connecté (équivalent à **who am i**).
-q	:	N'affiche que le nom et le nombre des utilisateurs connectés.
-u	:	Affiche le temps (heures minutes) d'inactivité de chaque utilisateur.
		. : Indique que l'utilisateur a été actif durant la dernière minute.
		old : Indique que l'utilisateur a été inactif depuis plus de 24 heures.
-w	:	Affiche le statut de l'utilisateur vis à vis des messages émis par la commande **write**.
		+ : Messages autorisés.
		- : Messages non autorisés.

Exemple

```
$ who -wuH
NOM         LIGNE        HEURE         OISIF          PID COMMENTAIRE
ae        + pts/3        Sep 13 12:28   .          15685
ae        + pts/0        Sep 12 11:35 03:16         2181 (10.0.3.68)
jcl       + pts/2        Sep 13 10:09 06:00        10617 (10.2.6.39)
```

Sous sa forme **whoami** la commande n'affiche que **l'utilisateur en cours ($USER)**.

III.2.2 id

Affichage des identifiants associés à l'utilisateur en cours.

Exemple

```
$ id
uid=513(ae) gid=501(formateur) groupes=501(formateur)
```

III.2.3 finger

Autre forme d'affichage d'**informations** sur les **utilisateurs connectés**.

Exemple

```
$ finger -l
Login: ae                               Name: Alain Escher
Directory: /home/formateur/ae           Shell: /bin/bash
On since Tue Sep 13 12:28 (CEST) on pts/3
On since Mon Sep 12 11:35 (CEST) on pts/0 from 10.0.3.68
   25 seconds idle
No mail.
No Plan.

Login: jcl                              Name: Jean-claude Lamant
Directory: /home/formateur/jcl          Shell: /bin/bash
On since Tue Sep 13 10:09 (CEST) on pts/2 from 10.2.6.39
   6 hours 12 minutes idle
No mail.
No Plan.
```

III.2.4 logname

Affichage du **nom de connexion**.

Exemple

```
$ logname
ae
```

III.2.5 tty

Affichage de la **voie de connexion** (nom du terminal).

Exemple

```
$ tty
/dev/pts/3
```

III.2.6 stty

Affichage des **caractéristiques du terminal**.

Options

-a	**:**	Affiche toutes les caractéristiques.

Exemple

```
$ stty -a
speed 38400 baud; rows 54; columns 168; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2 = <undef>; start
= ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W; lnext = ^V;
flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff -iuclc -ixany
-imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0 isig icanon
iexten echo echoe echok -echonl -noflsh -xcase -tostop
-echoprt echoctl echoke
```

III.3 Commandes de gestion du temps

III.3.1 date

Affichage de la date et heure système.

```
date "+chaine de caractères et directives de formatage"
```

Directives de formatage

%A : jour de la semaine en lettres.
%a : jour de la semaine en abrégé.
%B : mois en lettres.
%b : mois en lettres en abrégé.
%d : jour en chiffres.
%H : heure (de 00 à 23).
%j : jour de l'année (001 à 366).
%m : mois en chiffres.
%w : numéro du jour de la semaine (0 pour Dimanche à 6 pour Samedi).
%y : année sur 2 positions.
%Y : année sur 4 positions.
%Z : zone horaire.

Formats pré-définis

%c : équivalent à "+%a %m %Y %Z %T"
%D : équivalent à **"+ %m/%d/%y"**
%T : équivalent à **"+ %H:%M:%S"**

Caractères spéciaux

%% : pour afficher le caractère %
%n : pour provoquer un **saut de ligne**

Exemples

```
$ date
mer sep 14 12:02:02 CEST 2005

$ date "+Le %d/%m/%Y%nIl est %T"
Le 14/09/2005
Il est 12:10:12
```

III.3.2 cal

Affichage du **calendrier d'un mois ou d'une année**.

Exemple

```
    Octobre 2014
di lu ma me je ve sa
           1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

III.4 Commandes diverses

III.4.1 echo

Affichage de données (**chaîne de caractères** ou **contenu de variables**).

```
echo [-e] "chaine de caractères ... variables ... "
```

Options

-e	**:**	Pour interpréter les caractères de contrôle

Caractères de contrôles

\n	**:**	Pour générer un retour-chariot (new line)
\c	**:**	Pour empêcher le retour chariot

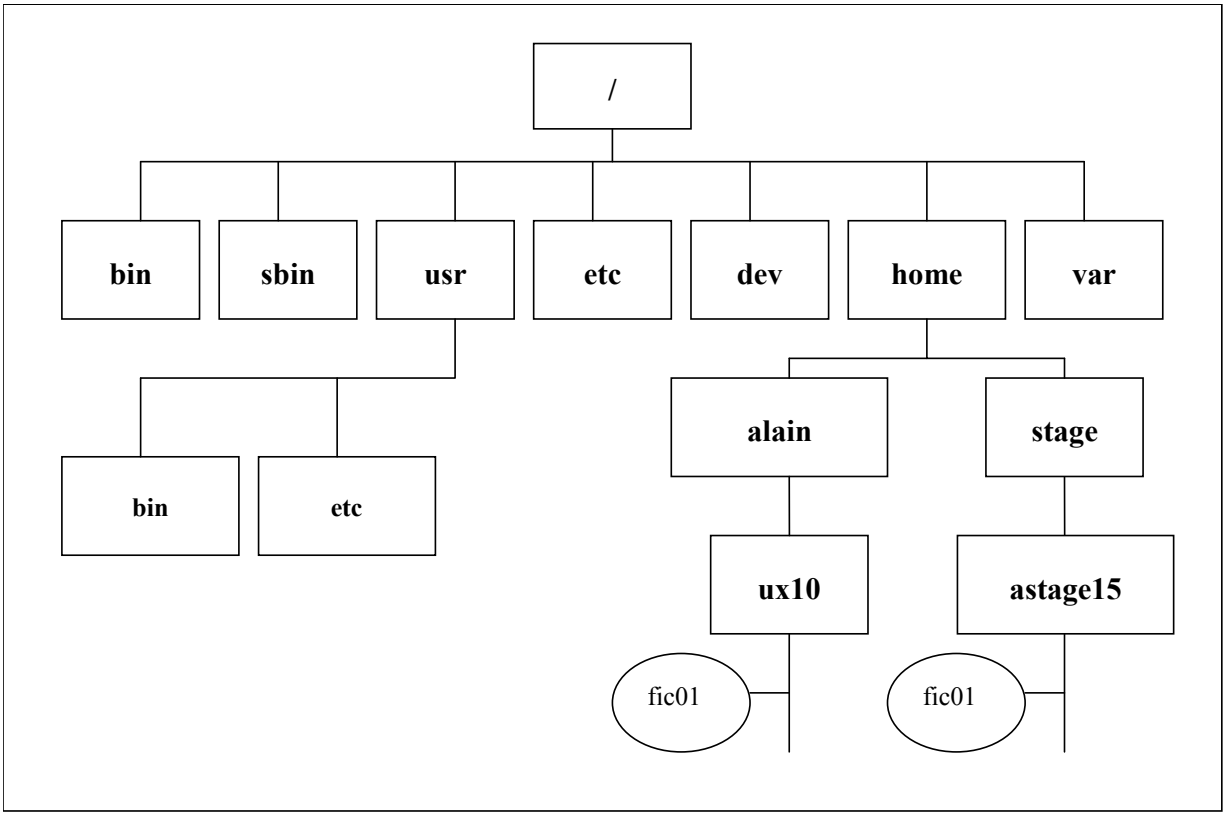
III.4.2 clear

Efface l'écran.

IV. Organisation des données d'un système Unix

IV.1 L'arborescence

Les données d'un système Unix forment un **ensemble de répertoires et de fichiers**, perçu par l'utilisateur comme une **arborescence "hiérarchique" unique**.



Cette arborescence est due à l'unification de différents **systèmes de fichiers** (ou "**partitions**") stockés sur un ou plusieurs disques.
La racine de chaque partition est attachée à un repertoire vide (ou **point de montage**) d'une autre partition.
Cette opération constitue le **montage** d'un système de fichier.

Un système Unix impose au moins 2 partitions :

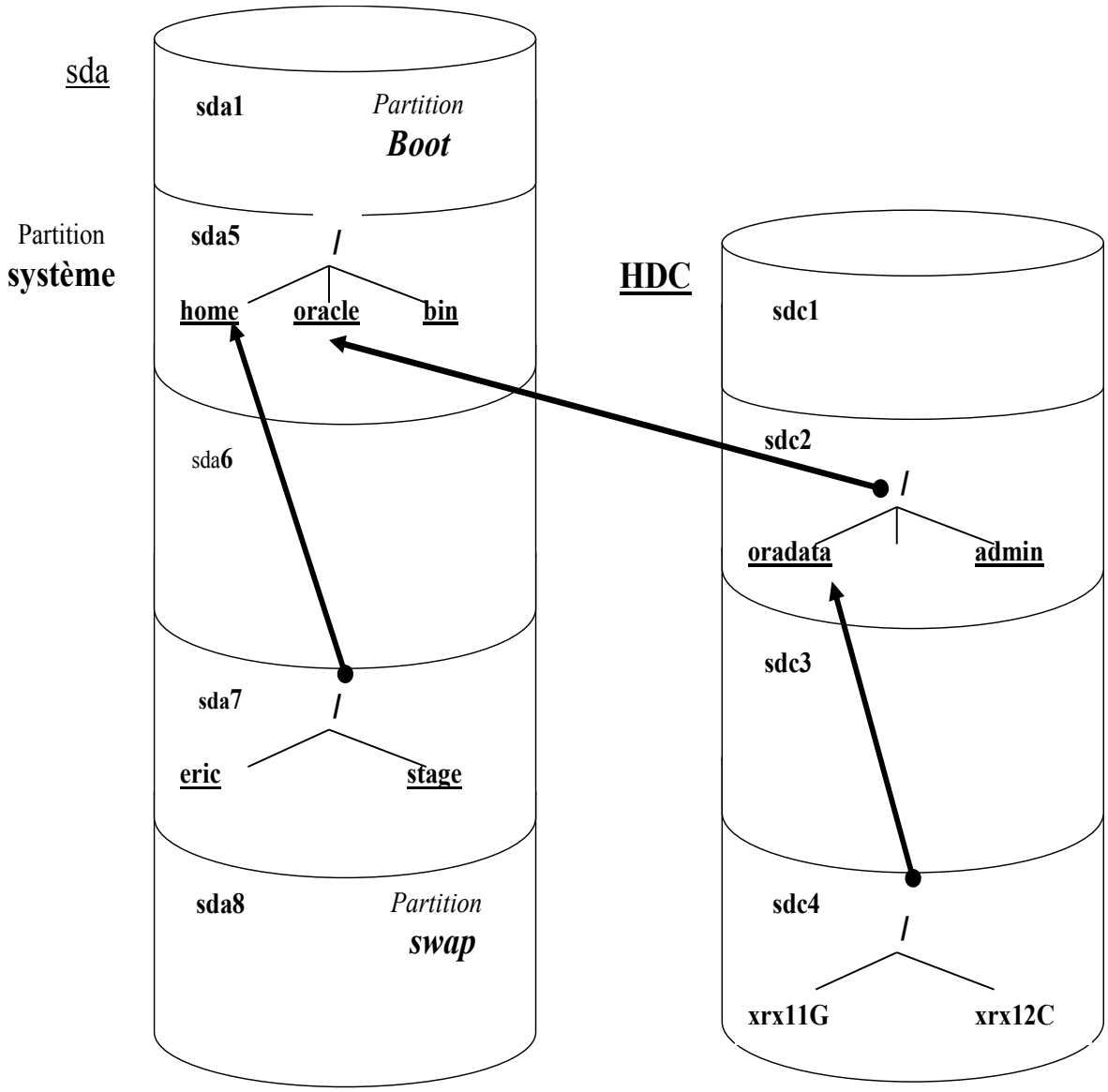
- La **partition principale** utilisée pour le stockage des données système. Cette partition est automatiquement montée lors du démarrage du système.
- La partition de "**swap**", utilisée pour la pagination.

Il est cependant conseillé de disposer d'au moins une partition supplémentaire, pour le stockage des **données utilisateur**.

IV.2 Organisation des partitions

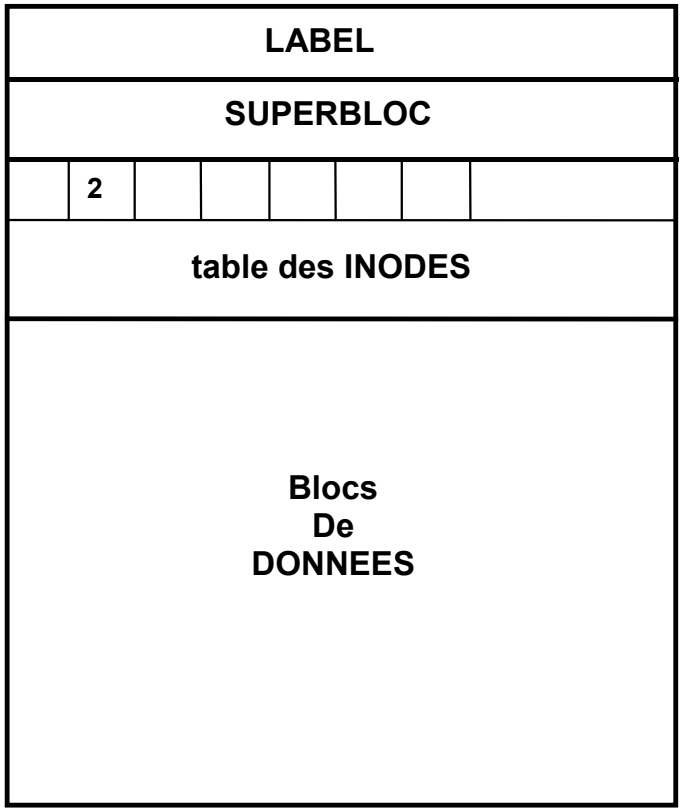
Les commandes suivantes sont utilisées :

- **fdisk** permet de **créer et gérer les partitions**. Le nom des fichiers spéciaux décrivant les disques et les partitions, varie selon le système Unix.
- **mkfs** permet de "**formater**" une partition. Le système de fichiers résultant, varie selon le système Unix.
- **mount** rend accessible le contenu de la partition.
  Le **montage** d'une partition consiste à connecter son contenu (c'est à dire une portion d'arborescence) à la partition système ou à une partition préalablement montée. Cette "connexion" s'effectue par l'intermédiaire du **point de montage**.



IV.3 Structure d'un système de fichiers

Un système de fichiers est constitué des éléments suivants.



- Le **label** qui contient le **nom de la partition** et l'**indicateur de montage**.

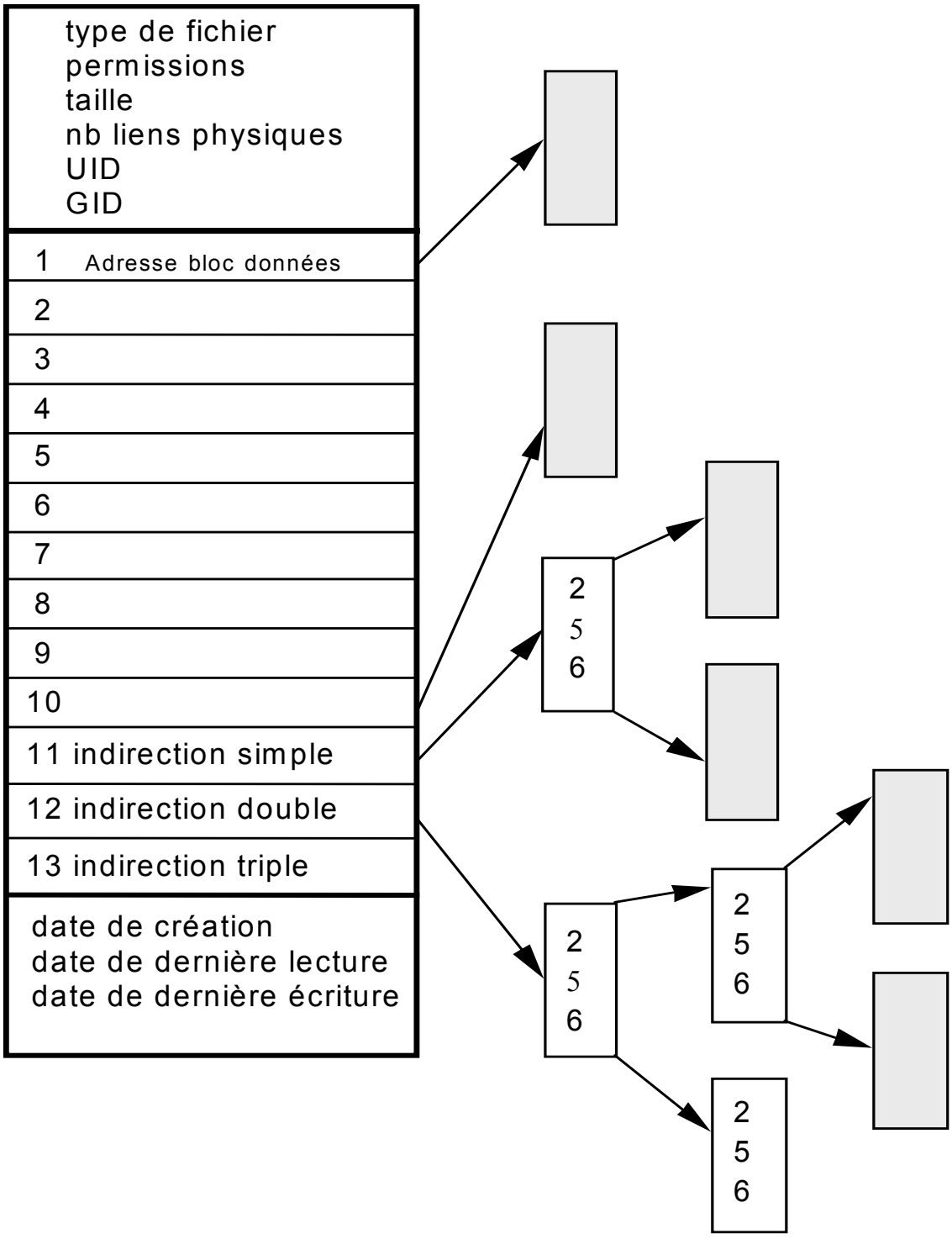
- Le **super-bloc** qui contient les caractéristiques de la partition :
  - Date de création
  - Taille en nombre de blocs et en nombre d'inodes
  - Liste des inodes disponibles
  - Liste des blocs de données disponibles

- La table des **inodes**.
  L'inode est une structure de données (**64** ou **128 octets**) contenant les **caractéristiques d'un fichier**, ainsi que l'adresse de chacun de ses blocs de données.
  L'inode 2 pointe sur la racine de la partition.

- Les **blocs de données**.

IV.3.1 Structure de l'inode

Un fichier est associé à une inode unique, qui lui est affectée lors de sa **création**.



IV.4 Arborescence système standard d'un système Unix

/	:	Répertoire **racine**. Il représente le point de départ de l'arborescence.
/bin	:	Répertoire des exécutables de la plupart des **commandes ou utilitaires Unix**.
/sbin	:	Répertoire référençant certaines commandes ainsi que de nombreux **fichiers d'administration**.
/usr	:	Contient des sous-répertoires liés à l'administration et aux fichiers publics (pages de manuel par exemple).
/etc	:	Répertoire des **fichiers de configuration**.
/dev	:	Répertoire des **fichiers spéciaux** associés aux périphériques.
/home	:	Répertoire à partir duquel se trouvent les **répertoires de connexion** des utilisateurs.
/var	:	Répertoire des "données variées". Il est utilisé, entre autres, pour les files d'attente et les boites aux lettres.
/lost+found	:	L'utilitaire **fsck** y place les "fichiers perdus" (inode non vide mais référencée dans aucun répertoire) produits pardes erreurs disques ou des arrêts "violents" du système. Ce répertoire existe dans chaque partition.

IV.5 Les commandes de gestion de partitions et d'arborescence

IV.5.1 La commande mount

```
mount système_de_fichier point_de_montage
```

Utilisée sans option ni argument, la commande affiche la liste des partitions montées.

Exemple

```
$ mount
/dev/sda8 on / type ext2 (rw)
none on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/sda1 on /boot type ext2 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sda6 on /home type ext2 (rw)
/dev/sda2 on /pourOracle type ext2 (rw)
none on /dev/shm type tmpfs (rw)
/dev/sda9 on /tmp type ext2 (rw)
/dev/sda3 on /usr type ext2 (rw)
/dev/sda5 on /var type ext2 (rw)
```

IV.5.2 la commande df

```
df [ nom_partition ]
```

Affichage d'informations concernant l'**espace occupé** et l'**espace disponible** d'une ou plusieurs **partitions**.

Options

-i	**:**	Affichage du nombre d'inodes au lieu du nombre de blocs.

Exemple

```
$ df /dev/sda[6-8]
SysFichier           1K-blocs    Utilisé Dispo.   Util% Monté sur
/dev/sda6              2030736    415724   1510192  22% /home
/dev/sda7              1011448     84384    874856   9%
/dev/sda8              1011448     84384    874856   9% /
```

IV.5.3 la commande du

```
du [ nom_répertoire … ]
```

Affichage de la taille de l'arborescence, à partir d'un répertoire.

Options

-a	**:**	Affichage de la taille des fichiers.
-s	**:**	N'affiche que le total.

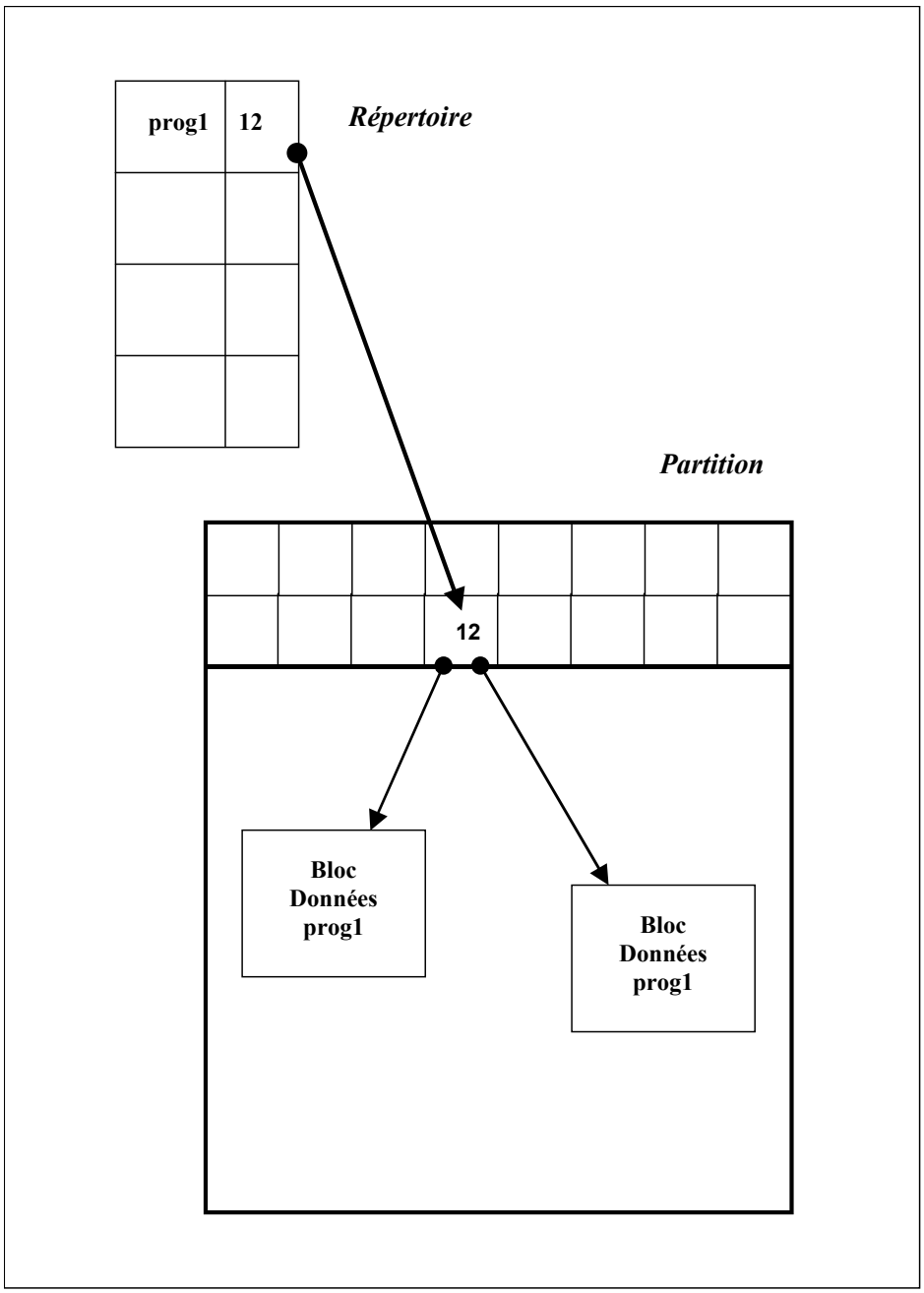
Exemple

```
$ du -a bin
4       bin/rnc
12      bin/cat_ae
12      bin/uptolo
32      bin
```

V. LES FICHIERS

V.1 Notion de fichier

Sous Unix, **un fichier est la plus petite unité d'information**.
C'est grâce au système de fichiers que l'on peut identifier (localiser) un fichier, quel que soit son type et son emplacement.
Un fichier est référencé dans un répertoire.
C'est le poste du répertoire qui permet de pointer sur l'inode affectée au fichier. L'inode contient les propriétés (caractéristiques) du fichier ainsi que les adresses des blocs de données.



V.2 Notion de chemin d'accès

V.2.1 Chemin d'accès absolu

Le chemin d'accès absolu d'un fichier, décrit l'itinéraire à emprunter depuis la **racine** (/) jusqu'au fichier.
On peut dire dans ce cas, que le nom absolu d'un fichier commence toujours par un /.
La longueur d'un chemin absolu est limitée à 1024 caractères.

V.2.2 Chemin d'accès relatif

Le chemin d'accès d'un fichier, décrit l'itinéraire à emprunter depuis le **répertoire en cours** jusqu'au fichier. On peut dire dans ce cas, que le nom relatif d'un fichier ne commence pas par un /.
Un chemin relatif peut contenir les caractères spéciaux suivants.

..	**:**	Représente le **répertoire père** (niveau supérieur dans l'arborescence).
.	**:**	Représente le **répertoire en cours.**
-	**:**	Représente le répertoire **précédent** (variable **$OLDPWD**).
~user	**:**	Représente le **répertoire de connexion** de l'utilisateur spécifié.

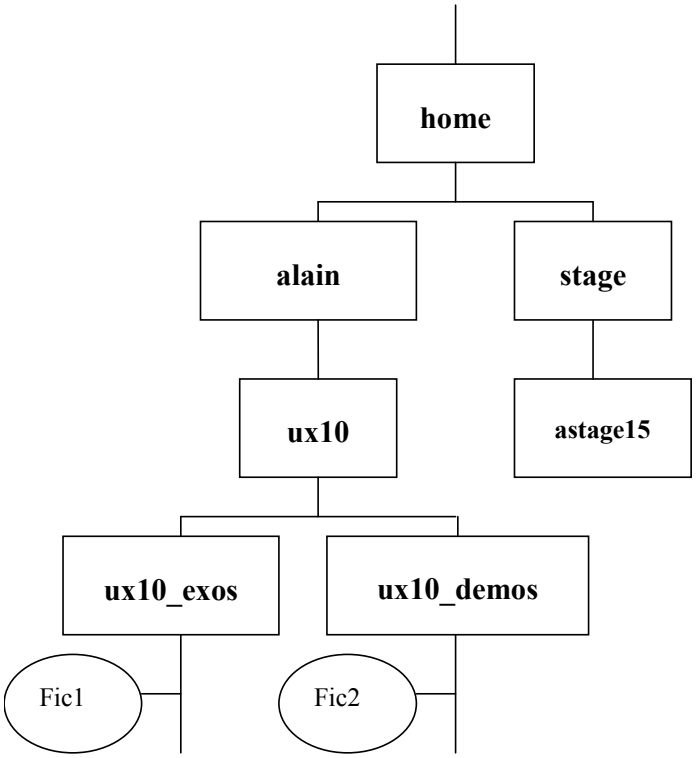
V.2.3 Les commandes dirname et basename

La commande **dirname** affiche le chemin d'accès à un fichier, depuis la **racine**.
La commande **basename** affiche nom d'un fichier.

Exemple

```
$ pwd
/home/stage/astage15
$ dirname $PWD
/home/stage
$ basename $PWD
astage15
```

Exemples d'expressions de chemin d'accès




```
$ pwd
/home/stage/astage15
$ ls -l /etc/passwd
-rw-r--r-- 1 root root 4959 janv.  6  2014 /etc/passwd
$ ls -l bin/fic1
-rw-r--r-- 1 astage15 stage 0 oct.  27 12:05 bin/fic1
$ cd bin
[ae@unix-cours bin]$ ls -l fic1
-rw-r--r-- 1 astage15 stage 0 oct.  27 12:05 fic1
[ae@unix-cours bin]$ ls -l ../src/fic2
-rw-r--r-- 1 astage15 stage 0 oct.  27 12:07 ../src/fic2
[ae@unix-cours bin]$ cd
$ ls -ld ~
drwx------ 6 astage15 stage 4096 oct.  27 12:07 /home/stage/astage15
```

V.3 Caractéristiques d'un fichier

La plupart des caractéristiques d'un fichier, peuvent être affichées grâca à l'option **–l** de la commande **ls**.

```
$ ls -lh
total 12K
drwxr-xr-x 2 astage15 stage 4,0K oct.  27 12:05 bin
lrwxrwxrwx 1 astage15 stage   16 oct.  27 12:22 data -> /home/stage/data
drwx------ 5 astage15 stage 4,0K oct.  24 20:00 Desktop
drwxr-xr-x 2 astage15 stage 4,0K oct.  27 12:07 src
-rw-r--r-- 1 astage15 stage    0 oct.  27 12:15 un_fichier
-rwxr--r-- 1 astage15 stage    0 oct.  27 12:15 un_script
$ ls -l /dev/fd0
brw-rw---- 1 root disk 8, 1 oct.   9 14:07 /dev/sda1
$ ls -l `tty`
crw--w---- 1 alain tty 136, 0 oct.  27 12:19 /dev/pts/0
```

Toutes ces caractéristiques, hormis la dernière (nom du fichier), sont stockées dans l'**inode** affectée au fichier.

- Premier champ : **Type de fichier**.
  - **d** : Pour les **répertoires** (**d**irectory).
  - **-** : Pour les fichiers **ordinaires**.
  - **l** : Pour les **liens symboliques**.
  - **c** : Pour les fichiers spéciaux associés aux périphériques qui traitent des **c**aractères (imprimantes, écrans).
  - **b** : Pour les fichiers spéciaux associés aux périphériques qui traitent des **b**locs (disques, disquettes…).
- Deuxième champ : Les **privilèges** (9 caractères).
- Troisième champ : Nombre de **liens physiques** associés au fichier.
- Quatrième champ : Nom du **propriétaire** du fichier.
- Cinquième champ : Nom du **groupe** dont le propriétaire est membre.
- Sixième champ : **Taille** du fichier (exprimée en octets).
- Septième champ : Date de **dernière utilisation** du fichier.
- Huitième champ : Nom du fichier (limité à 255 caractères).

V.3.1 La commande file

Cette commande permet également de connaître la **nature** d'un ou de plusieurs **fichiers**.

```
file nom_fichier …
```

V.4 Les métacaractères de nom de fichiers

Les métacaractères sont des **caractères de substitution**, que l'on utilise dans les noms de fichiers.
Ils sont traîtés par le **shell** avant l'exécution de la commande.
Le nombre de métacaractères utilisés dans un nom de fichier, n'est pas limité.

- ***** : Remplace de 0 à n caractères, sauf le / et le . en début de nom de fichiers.
- **?** : Remplace 1 caractère, sauf le / et le . en début de nom de fichiers.
- **[xy]** : Remplace les caractères spécifiés dans la liste.
- **[!xy]** : Remplace les caractères non spécifiés dans la liste.
- **[x-y]** : Remplace les caractères compris entre les deux caractères spécifiés.
- **[!x-y]** : Remplace les caractères non compris entre les deux caractères spécifiés.

V.4.1 Protection des métacaractères

Il existe différentes méthodes permettant de faire perdre sa signification à un métacaractère, de manière à ce qu'il soit normalement transmis à la commande.

- \ : Caractère d'échappement, qui annule la signification du caractère suivant.
- ' : Encadre une chaîne de caractères dont les métacaractères sont protégés.
- " : Encadre une chaîne de caractères dont les métacaractères sont protégés, sauf **$**, ` et \ qui conservent leur signification.

```
$ ls fica*
fica  ficaa  ficaaa  ficab  ficac
$ ls fica*b
ficab
$ ls *ich*
fichier1  fichier2
$ ls *a*a*
demo_metacaracteres  ficaa  ficaaa
$ ls fica?
ficaa  ficab  ficac
$ ls fic???
ficaaa
$ ls fic[ab]
fica  ficb
$ ls fic[ab]*
fica  ficaa  ficaaa  ficab  ficac  ficb  ficbb
$ ls fic[ab]?*
ficaa  ficaaa  ficab  ficac  ficbb
$ ls fic[!ab]?*
fic01  fic03  fic18  fic29  ficcc  fichier1  fichier2
$ ls fic[0-9]
fic1
$ ls fic[0-9][0-9]
fic01  fic03  fic18  fic29
$ ls fic[!0-9][!0-9]
ficaa  ficab  ficac  ficbb  ficcc
```

V.5 COMMANDES DE GESTION DES REPERTOIRES

V.5.1 Création de répertoire : mkdir

```
mkdir [-p] nom_répertoire …
```

Création des répertoires dont les noms (absolus ou relatifs) sont spécifiés.

Options

-p	**:**	Pour créer les éventuels répertoires pères, s'ils n'existent pas.

Exemple

```
$ mkdir -v rep1
mkdir: création du répertoire 'rep1'

$ mkdir -vp rep2/rep3
mkdir: création du répertoire 'rep2'
mkdir: création du répertoire 'rep2/rep3'
```

V.5.2 Supression de répertoire : rmdir et rm

```
rmdir nom_répertoire …
rm -r nom_répertoire …
```

Suppression des répertoires dont les noms (absolus ou relatifs) sont spécifiés :

- **rmdir** pour un répertoire **vide**.
- **rm –r** pour un répertoire **non vide**.

V.5.3 Autres commandes : cd et pwd

La commande **cd** permet de **modifier le répertoire en cours**.
Le nom du répertoire précédent est stocké dans la variable **$OLDPWD** (accessible grâce au caractère – dans un chemin d'accès relatif).

La commande **pwd** permet d'**afficher le répertoire en cours**.

V.6 LES COMMANDES DE GESTION DE FICHIERS

V.6.1 Listes de fichiers : La commande ls

```
ls [ options ] [ argument … ]
```

Selon le type d'argument spécifié :
- Affiche le **contenu du répertoire**.
- Affiche le **nom du fichier**.

-l	:	Affiche les **caractéristiques des fichiers**.
-a	:	Affiche les **fichiers cachés** (dont le nom commence par un point).
-i	:	Affiche le numéro de l'**inode** associée au fichier.
-R	:	Affiche le **contenu des sous-répertoires**.
-r	:	Tri inverse sur l'ordre alphabétique.
-d	:	N'affiche que le nom d'un répertoire, sans en afficher le contenu.
-t	:	Tri sur la date au lieu de l'ordre alphabétique.
-F	:	Ajoute un caractère à la fin du nom de certains fichiers, afin d'identifier leur type (/ pour un répertoire, * pour un exécutable et @ pour un lien symbolique).
-1	:	Affiche un nom de fichier par ligne.

```
Affichage du contenu du répertoire en cours
$ ls -F
bin/  data@  Desktop/  src/  un_fichier  un_script*

Affichage du contenu de plusieurs répertoires
$ ls bin src
bin:
fic1

src:
fic2

Affichage des caractéristiques des fichiers d'un répertoire
$ ls -l bin
total 0
-rw-r--r-- 1 astage15 stage 0 oct.  27 12:05 fic1

Affichage des caractéristiques de plusieurs répertoires
$ ls -ld bin src
drwxr-xr-x 2 astage15 stage 4096 oct.  27 12:05 bin
drwxr-xr-x 2 astage15 stage 4096 oct.  27 12:07 src
```

V.6.2 Mise à jour des dates de fichiers : La commande touch

```
touch -trac nom_fichier …
```

Mise à jour de la date de modification ou de dernier accès au fichier.

- **Par défaut**, la date en cours devient la date de dernière modification du fichier.
- Si le fichier n'existe pas, il est créé.

Options

-d	**:**	Pour affecter une valeur autre que la date en cours (format ***[[SS]AA]MMJJhhmm[.ss]***)
-r	**:**	Pour utiliser la date d'un fichier de référence.
-a	**:**	Pour modifier la date de dernier accès au lieu de la date de dernière modification.
-c	**:**	Pour ne pas créer le fichier s'il n'existe pas.

Exemple

```
$ touch fic_current
$ touch -d 20140101 fic_autre
$ touch -r Desktop fic_fic
$ ls -ld fic* Desktop
drwx------ 5 astage15 stage 4096 oct.  24 20:00 Desktop
-rw-r--r-- 1 astage15 stage    0 janv.  1  2014 fic_autre
-rw-r--r-- 1 astage15 stage    0 oct.  27 15:38 fic_current
-rw-r--r-- 1 astage15 stage    0 oct.  24 20:00 fic_fic
```

V.6.3 Affichage du contenu de fichier(s) : la commande cat

```
cat nom_fichier …
```

La commande **cat** affiche le contenu d'un ou de plusieurs fichiers sur la **sortie standard** (canal 1).
Si aucun nom de fichier n'est spécifié, les données à afficher seront lues sur l'entrée standard (canal 0).

V.6.4 Suppression de fichier(s) : la commande rm

```
rm -vrif nom_fichier …
```

La commande **rm** supprime un ou plusieurs fichiers.

Options

-v	:	Pour afficher la **liste des fichiers** supprimés.
-r	:	Pour supprimer le contenu d'un ou plusieurs **répertoires non vides**.
-i	:	Pour afficher une **demande de confirmation** de suppression, pour chaque fichier à supprimer.
-f	:	Pour ne pas déclencher d'erreur si le fichier n'existe pas.

V.6.5 Copie de fichiers : la commande cp

```
(1) cp -ip  fichier_origine     fichier_cible
(2) cp -ip  fichier(s)_origine répertoire_destination
(3) cp -rip répertoire_origine répertoire_destination
```

La commande **cp** permet :

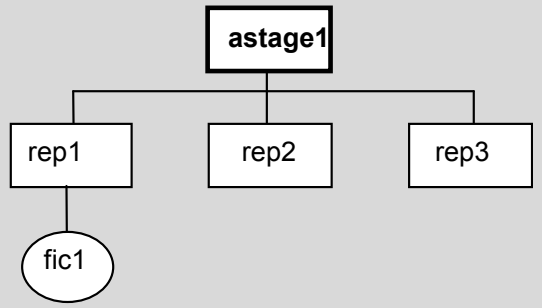
- **(1)** De copier un fichier dans un autre fichier.
  Si le fichier cible n'existe pas, il est créé.
  Si le fichier cible existe, il est détruit puis re-créé.
- **(2)** De copier un ou plusieurs fichiers sous un répertoire.
  La destination, dans ce cas, doit obligatoirement être un répertoire existant.
- **(3)** De copier une portion d'arborescence sous un répertoire.

Options

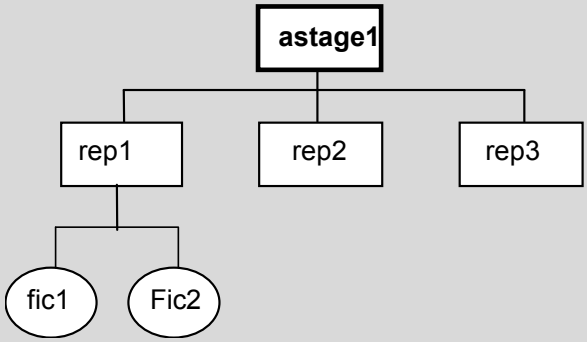
-i	:	Pour demander la confirmation de suppression des fichiers cibles existant.
-p	:	Pour copier les droits et la date de dernière modification des fichiers origines.
-r	:	Pour une copie d'arborescence (répertoire vers répertoire).

Exemples

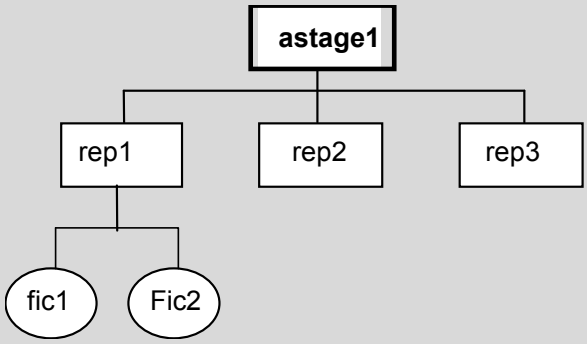
Copie d'un fichier dans un fichier



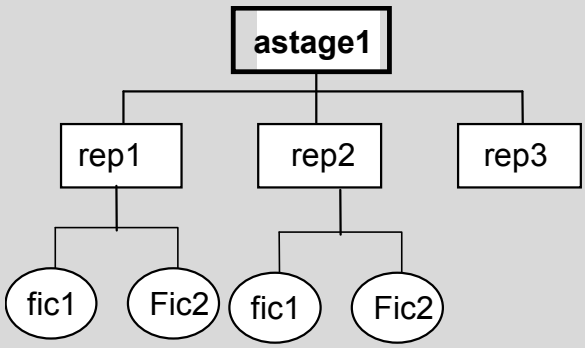

```
$ cp rep1/fic1 rep1/fic2
```



Copie de plusieurs fichier sous un répertoire

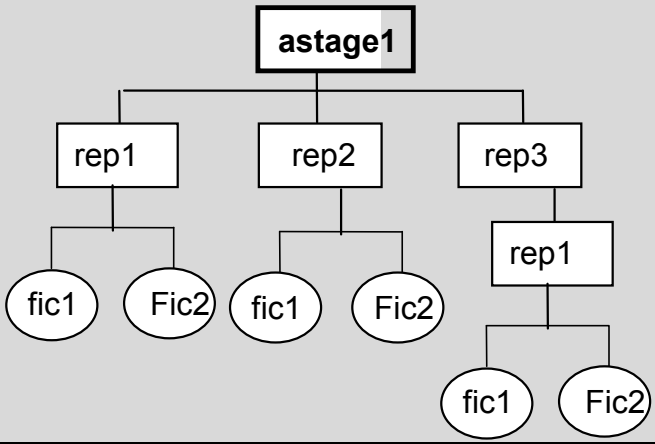



```
$ cp rep1/* rep2
```



Copie d'arborescence

```
$ cp -r rep1 rep3
```



V.6.6 Déplacement de fichiers : la commande mv

```
(1) mv -if fichier_origine     fichier_cible
(2) mv -if fichier(s)_origine répertoire_destination
(3) mv -if répertoire_origine répertoire_destination
```

La commande **mv** permet :

- **(1)** De déplacer/renommer un fichier.
- **(2)** De déplacer un ou plusieurs fichiers.
  La destination, dans ce cas, doit obligatoirement être un répertoire existant.
- **(3)** De renommer un répertoire ou de déplacer une portion d'arborescence.

Options

-i	**:**	Pour demander la confirmation de suppression des fichiers cibles existants.
-f	**:**	Pour écraser les fichiers de destination existants sans demande de confirmation.

V.7 Les liens

V.7.1 Création de liens physiques

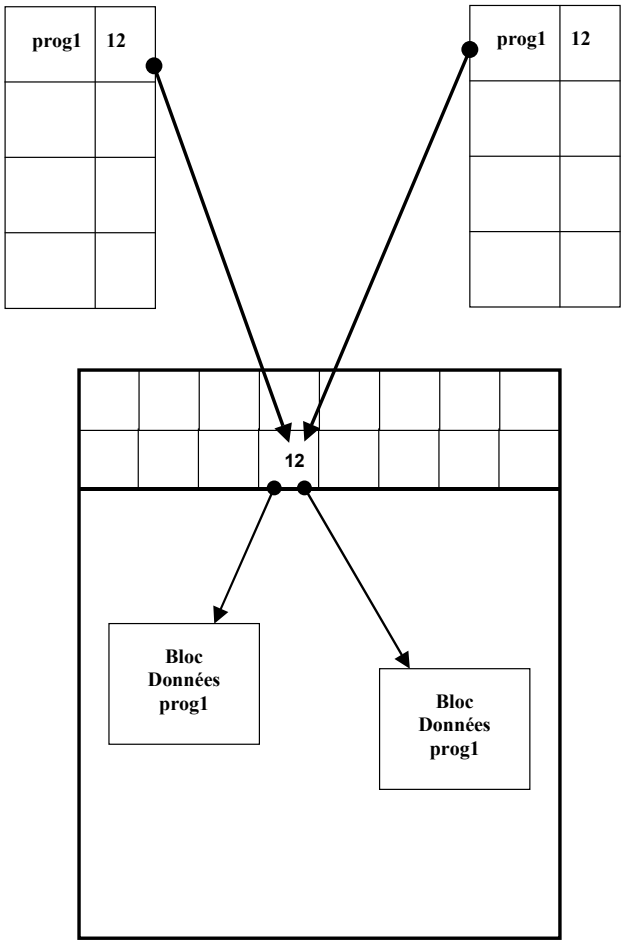
Créer un **lien physique** (ou matériel), consiste à associer un autre nom à un fichier déjà existant. Le nombre de liens physiques pointant sur un fichier est stocké dans l'**inode** de celui-ci.
Un lien physique permet donc à un utilisateur d'accéder un fichier grâce à un nom mémorisé dans l'un de ses répertoires, tout en lui évitant de manipuler des chemins d'accès complexes.

```
ln nom_fichier [ nom_lien ]
```

- Le fichier et son lien doivent se trouver dans le **même système de fichiers**.
- **Par défaut**, le lien portera le nom du fichier lié.
- Les liens physiques sont interdits sur les répertoires.

Exemple

```
$ ln bin/prog1
```



V.7.2 Création de liens symboliques

Créer un **lien symbolique** (ou matériel), consiste à associer un autre nom à un **répertoire** ou à un fichier résidant dans autre système de fichiers.
Le lien symbolique est un fichier à part entière (de type **l**), associé à un inode qui lui est propre et dont l'unique donnée est le chemin d'accès du fichier lié.

```
ln -s nom_répertoire [ nom_lien ]
ln -s nom_fichier    [ nom_lien ]
```

- **Par défaut**, le lien portera le nom du répertoire ou du fichier lié.

Exemple

```
$ ln -s ~astage15 lnk_astage15
```

V.8 Recherche de fichier : la commande find

```
find rép_départ [ critère_sélection … ] [critère_exécution ]
```

La commande **find** cherche et affiche sur la sortie standard, les fichiers dont le nom correspond au(x) critère(s) spécifié(s).

- *repertoire_départ* identifie le **répertoire à partir duquel** la recherche démarre.

Principaux critères de sélection

-name	:	Nom relatif des fichiers à chercher. La chaîne peut contenir des métacaractères. Elle doit dans ce cas être saisie entre quotes.
-type	:	Spécifie le type de fichiers à chercher.
d	:	Pour des **répertoires**.
f	:	Pour des fichiers **ordinaires**.
l	:	Pour des **liens symboliques**.
b	:	Pour des fichiers spéciaux de type blocs.
c	:	Pour des fichiers spéciaux de type caractères.
p	:	Pour des fichiers "pipe".
-inum	:	Spécifie le numéro d'**inode** des fichiers à chercher.
-user	:	Spécifie le propriétaire des fichiers à chercher.
-perm	:	Spécifie les droits (en octal) sur les fichiers à chercher.
-size	:	Spécifie la taille des fichiers à chercher.
-atime	:	Pour chercher des fichiers qui n'ont pas été accédés depuis ***n*** jours.
-mtime	:	Pour chercher des fichiers qui n'ont pas été modifiés depuis ***n*** jours.
-ctime	:	Pour chercher des fichiers qui ont été créés il y a ***n*** jours.

Lorsque **plusieurs critères** sont spécifiés, ils sont implicitement liés par un **et logique**.
L'option **–o** permet de les lier par un **ou logique**.
La négation est spécifiée en plaçant un **!** devant le critère.

Critères d'exécution

-exec	:	Commande à exécuter. Les caractères {} représentent le nom du fichier.
-ok	:	Demande de confirmation de la commande à exécuter.
-print	:	Pour afficher le résultat de la recherche.

Exemples

```
$ find . -name 'ex*'
./examples.desktop
./exo
./exo/exo1
./ux15/ux15_demos/exercices
./exercices
$ find . -maxdepth 1 -name 'ex*'
./examples.desktop
./exo
./exercices
```

Exemples (find suite)

```
$ find exo -type f -name 'e*'
exo/essai2

$ find exo -type f \( -name 'e*' -o -name 'g*' \)
exo/exo1/group
exo/group
exo/essai2

$ find . -name 'ex*' -exec ls -ld {} \;
-rw-r--r-- 1 alain alain 8942 nov.  22  2013 ./examples.desktop
drwxrwxr-x 3 alain alain 4096 oct.  27 16:25 ./exo
drwxrwxr-x 2 alain alain 4096 janv.  7  2014 ./exo/exo1
drwxrwxr-x 2 alain alain 4096 juin  26 11:56 ./ux15/ux15_demos/exercices
drwxrwxr-x 2 alain alain 4096 nov.  22  2013 ./exercices
```

V.9 Comparaison de fichiers

V.9.1 La commande diff

La commande **diff** compare deux répertoires ou deux fichiers.

```
diff fichier1 fichier2
diff repertoire1 repertoire2
```

Options

-b	**:**	Pour ignorer les différences dues aux caractères "blancs".

Exemple

```
[ae@unix-cours2 ux15]$ diff xec xec2
5c5
< chmod u+x $1
---
> chmod 744 $1
```

V.9.2 La commande cmp

La commande **cmp** fait une comparaison binaire de 2 fichiers.

V.9.3 La commande comm

La commande **comm** compare ligne à ligne deux fichiers triés.

```
comm [ options ] fichier1 fichier2
```

Sans option, la commande affiche 3 colonnes :

- La colonne de gauche contient les lignes présentes uniquement dans ***fichier1***.
- La colonne du milieu contient les lignes présentes uniquement dans ***fichier2***.
- La colonne de droite contient les lignes communes aux 2 fichiers.

Options

-1	**:**	Pour supprimer l'affichage des lignes présentes uniquement dans ***fichier1***.
-2	**:**	Pour supprimer l'affichage des lignes présentes uniquement dans ***fichier2***.
-3	**:**	Pour supprimer l'affichage des lignes communes.

VI. Redirection des entrées et sorties standards

VI.1 CANAUX D'ENTREES ET SORTIES STANDARDS

Tout processus dispose de trois "canaux" lui permettant de traiter des flux de données.

- Un **canal d'entrée** (canal **0**) permettant la **lecture de données**.
  Ce canal est associé par défaut au **clavier**.
- Un **canal de sortie** (canal **1**) permettant l'**écriture des données** résultantes.
  Ce canal est associé par défaut à l'**écran**.
- Un **canal de sortie** (canal **2**) permettant l'**écriture des messages d'erreurs**.
  Ce canal est associé par défaut à l'**écran**.

Le mécanisme dit de "redirection", permet de remplacer les fichiers périphériques standards par d'autres fichiers ordinaires ou spéciaux.

VI.2 Redirection de SORTIE

```
commande n°_canal> fichier_sortie
```

Pour rediriger l'un des canaux de sortie vers un fichier ordinaire ou spécial.

- Par **défaut** : canal **1**.
- Chaque canal de sortie peut être redirigé vers un fichier.
- Le caractère **&** placé à gauche du métacaractère de redirection, permet de rediriger les **canaux 1 et 2** vers un même fichier de sortie.
- Lorsqu'un **canal doit être redirigé vers l'autre canal**, le nom du fichier de sortie doit être remplacé par ***&n°_canal***.

Métacaractères de redirection de sortie

>	:	Créé le fichier de sortie, s'il n'existe pas. Remplace le fichier de sortie, s'il existe déjà.
>>	:	Créé le fichier de sortie, s'il n'existe pas. Complète le fichier de sortie, s'il existe déjà.

Exemples

```
# Redirection du résultat d'une commande dans un fichier ordinaire
$ date "+Utilisateurs connectes le %d/%m/%Y" > compte_rendu

# Mise à jour du fichier ordinaire de sortie
$ who >> compte_rendu

# Redirection du résultat et des erreurs
$ find /home -inum 16384 > fic 2> /dev/null

# Redirection du résultat et des erreurs vers un même fichier
$ find /home -inum 16384 &> fic

# Redirection du canal 1 vers le canal 2
$ echo "Erreur - Arrêt du processus" >&2
```

VI.3 Redirection d'ENTREE

```
commande 0< fichier_entrée
```

Pour rediriger le canal d'entrée vers un fichier ordinaire.

- Par **défaut** : canal **0**.

Métacaractères de redirection de sortie

<	:	Permet à la commande de lire le fichier spécifié à droite du métacaractère.

Exemples

```
Lecture d'un fichier ordinaire par une commande
$ cat < un_fichier > un_autre_fichier
```

VII. LES FILTRES

VII.1 Utilisation des filtres – Mécanisme du "pipe"

Ce mécanisme consiste en l'exécution d'une commande, dont le résultat et lu et traité par une autre commande.

```
commande | filtre …
```

Les règles suivantes doivent être respectées :

- La première commande de l'ensemble doit produire un résultat sur son **canal 1**.
- Les commandes intermédiaires et la dernière commande doivent être des filtres, c'est à dire des commandes Unix utilisant leur **canal 0**.

VII.2 head et tail : Affichage de lignes

```
head/tail [ -options ] nom_fichier
commande | head/tail [ -options ]
```

Ces filtres permettent d'afficher les n premières lignes (**head**) ou les n dernières lignes (**tail**) d'un fichier ou du résultat de la commande connectée à l'entrée du "pipe".

Options

-n	:	Pour afficher *n* lignes (par défaut **10**).
-q	:	Le résultat n'est pas affiché.
-v	:	Pour afficher les noms de fichiers.

Exemples

```
$ head -2v /etc/passwd /etc/group
==> /etc/passwd <==
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh

==> /etc/group <==
root:x:0:
daemon:x:1:

$ ps -ef | head -2
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 oct.09 ?       00:00:00 /sbin/init
```

VII.3 more et less : Pagination de fichiers

```
more/less [ -options ] nom_fichier
commande | more/less [ -options ]
```

Ces filtres permettent d'afficher le contenu d'un fichier ou d'une commande, page par page.
Le filtre **less**, plus récent et complet que **more**, propose des possibilités supplémentaires de pagination et de recherche de chaîne de caractères.

VII.4 Comptage d'éléments : wc

```
wc [ -lwc ] nom_fichier
commande | wc [ -lwc ]
```

Le filtre **wc** permet de compter les nombres de caractères, de mots et de lignes :

- Dans le fichier spécifié en tant qu'argument.
- Dans le résultat de la commande connectée à l'entrée du "pipe".

Le résultat est écrit sur le canal de sortie standard.
Si aucune option est spécifiée, les 3 statistiques sont affichées dans l'ordre suivant : nombre de lignes, de mots et de caractères.

Options

-l	:	Pour afficher le nombre de lignes.
-w	:	Pour afficher le nombre de mots.
-c	:	Pour afficher le nombre de caractères.

Exemples

```
$ wc -l /etc/passwd
99 /etc/passwd

$  ps -e h | wc -l
120
```

VII.5 grep : Recherche de chaine de caractères

```
grep [ -options ] chaîne nom_fichier
commande | grep [ -options ] chaîne
```

Le filtre **grep** **cherche une chaine de caractères** :

- Dans le fichier spécifié en tant qu'argument.
- Dans le résultat de la commande connectée à l'entrée du "pipe".

Le résultat est écrit sur le canal de sortie standard.

Options

-q	:	Le résultat n'est pas affiché.
-i	:	Pour annuler la distinction minuscules/majuscules.
-v	:	Pour afficher les lignes qui **ne contiennent pas** la chaîne de caractères.
-w	:	Pour afficher les lignes qui contiennent la chaîne de caractères sous forme d'un mot complet (un mot est constitué de lettres, chiffres et du souligné _).
-c	:	Pour n' afficher que le nombre de lignes contenant la chaîne de caractères.

Symboles utilisés dans les expressions régulières

'^chaine'	:	Pour rechercher la chaîne en **début de ligne**.
'chaine$'	:	Pour rechercher la chaîne en **fin de ligne**.
*'ch*ine'*	:	Remplace de **0 à n occurrences du caractère précédent**.
'ch+ine'	:	Remplace de **1 à n occurrences du caractère précédent**.
'ch.ine'	:	Remplace **un caractère** quelconque.
'c1|c2'	:	Pour rechercher **c1 ou c2** (**option –E** ou utilisation de **egrep** indispensables).

Exemples

```
$ grep '^root' /etc/passwd
root:x:0:0:root:/root:/bin/bash

$ grep -E '^(alain|eric)' /etc/passwd
eric:x:1000:1000:eric,,,:/home/eric:/bin/bash
alain:x:1061:1061::/home/alain:/bin/bash

$  ps -ef | grep 'pts/0'
astage15  3837  3716  0 10:23 ?        00:00:00 sshd: astage15@pts/0
astage15  3838  3837  0 10:23 pts/0    00:00:00 -bash
astage15  3975  3838  0 11:14 pts/0    00:00:00 ps -ef
astage15  3976  3838  0 11:14 pts/0    00:00:00 grep pts/0
```

VII.6 tr : Substitution de caractères

```
tr [ -options ] 'chaine_1' [ 'chaine_2' ]
commande | tr [ -options ] [ 'chaine_1' ] [ 'chaine_2' ]
```

Le filtre **tr** permet :

- De remplacer chaque caractère de '***chaine_1***' par son homologue dans '***chaine_2***'.
- De supprimer les caractères de '***chaine_1***', si l'option **–d** est spécifiée.
- De réduire à un caractère unique, les multilples occurrences contigues d'un caractère de '***chaine_1***', si l'option **–s** est spécifiée.
- **[:upper:]** représente l'ensemble des lettres majuscules.
- **[:lower:]** représente l'ensemble des lettres minuscules.

Exemples

```
$ grep -E '^(alain|eric)' /etc/passwd | cut -d: -f1 | tr '[a-z]' '[A-Z]'
ERIC
ALAIN

$ who am i | tr -s ' '
astage15 pts/0 2014-10-28 10:23 (10.236.226.179)
```

VII.7 cut : Extraction de colonnes

Le filtre **cut** permet d'extraire :

- Les caractères dont la position est spécifiée par l'option **–c**.
- Les colonnes (ou champs) dont le rang est spécifiée par l'option **–f**.
  Il convient dans ce cas, de définir un délimiteur de champ grâce à l'option **–d**.
  Le délimiteur par défaut est le caractère de **tabulation**.

```
cut -cn[,m,…] nom_fichier
cut -fn[,m,…] -d'délimiteur' nom_fichier
commande | cut [ -options ]
```

Identifications des caractères ou des colonnes à extraire

n	:	Pour extraire le nième caractère ou colonne.
n-m	:	Pour extraire les caractères ou colonnes compris entre les positions **n** et **m**.
-m	:	Pour extraire les caractères ou colonnes compris entre les positions **1** et **m**.
n-	:	Extraction de caract./colonnes compris entre la position **n** et le dernier caract./colonne.

Exemples

```
$ cut -d: -f1,3,4 /etc/passwd | tail -3
postfix:117:125
alain:1061:1061
guest-AeKBdR:118:127

$ ps -f --no-headers | tr -s ' ' | cut -d' ' -f1,8
astage15 -bash
astage15 ps
astage15 tr
astage15 cut
```

VII.8 sort : Tri

Le filtre **sort** permet :

- De trier les lignes du fichier spécifié en tant qu'argument.
- De trier les lignes du résultat de la commande connectée à l'entrée du "pipe".
- De fusionner les lignes de fichiers déjà triés.

```
sort [ -options ] fichier_1 [fichier_2 ]
commande | sort [ -options ]
```

Options

-b	:	**Réduit à un caractère unique, les multilples occurrences du caractère 'blanc'**.
-t	:	Définit le **délimiteur** de champs (**par défaut** : **blanc**).
-n	:	Pour trier sur un champ numérique (**-b** obligatoire dans ce cas)
-m	:	Fusion sur fichiers triés.
-u	:	Pour éliminer les doublons.
-f	:	Pas de distinction minuscules/majuscules.
-r	:	Pour trier en ordre décroissant.
-o	:	Pour stocker le résultat dans un fichier (revient à rediriger le canal 1 vers un fichier).

Définition des critères de tri

-k*x*,*y*	:	Pour définir un critère de tri. *x* réprésente le numéro du champ constituant le début du critère (à partir de **1**). *y* réprésente le numéro du champ constituant la fin du critère.
+*x*	:	*x* réprésente le numéro du champ constituant le début du critère (à partir de **0**).
-*y*	:	*y* représente le numéro du champ constituant la fin du critère**.**

Exemples

```
$ ps -ef | sort -bn -k3,3 -k2,2 | head
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 oct.09 ?       00:00:00 /sbin/init
root         2     0  0 oct.09 ?       00:00:00 [kthreadd]
root       301     1  0 oct.09 ?       00:00:00 upstart-file-bridge --daemon
root       458     1  0 oct.09 ?       00:00:00 upstart-udev-bridge --daemon
root       471     1  0 oct.09 ?       00:00:00 /usr/sbin/sshd -D
root       474     1  0 oct.09 ?       00:00:00 /sbin/udevd --daemon
102        481     1  0 oct.09 ?       00:00:07 dbus-daemon --system --fork
syslog     537     1  0 oct.09 ?       00:01:26 rsyslogd -c5
root       646     1  0 oct.09 ?       00:00:00 upstart-socket-bridge --daemon
```

VII.9 Elimination de doublons : le filtre uniq

```
uniq  [ -options ] fic_a_traiter fic_de_sortie
```

Le filtre **uniq** permet d'éliminer des lignes consécutives identiques :
- Du fichier spécifié en tant qu'argument.
- Du résultat de la commande connectée à l'entrée du "pipe".

Option

-u	**:**	N'affiche que les lignes ne présentant pas de doublons.
-d	**:**	N'affiche que les lignes présentant des doublons.
-c	**:**	Affiche chaque ligne précédée du nombre de lignes identiques.

Arguments

*fic_a_trait*er	**:**	Nom du fichier en entrée.
fic_de_sortie	**:**	Nom du fichier de sortie.

Exemples

```
$ cat prenoms
eric
eric
martial sans doublon
alain
alain
eric sans doublon
martial sans doublon
$ uniq prenoms
eric
martial sans doublon
alain
eric sans doublon
martial sans doublon
$ uniq -u prenoms
martial sans doublon
eric sans doublon
martial sans doublon
$ uniq -d prenoms
eric
alain
$ uniq -c prenoms
      2 eric
      1 martial sans doublon
      2 alain
      1 eric sans doublon
      1 martial sans doublon
```

VII.10 split : Découpage de fichiers

```
split -n fic_a_déc fic_de_sor
```

Le filtre **split** permet de **scinder un fichier en plusieurs**, en fonction d'un nombre de lignes.

Option

-n	**:**	Nombre de lignes de chaque fichier.

Arguments

fic_a_déc	**:**	Nom du fichier en entrée (fichier à scinder).
fic_de_sor	**:**	Nom des fichiers de sortie. Chaque nom sera suffixé par '**aa**', '**ab**' etc…

Exemples

```
$ cat pair
ligne8 8
ligne8 8
ligne6 6
lOgne6 6
ligne8 8
$ split -3 pair out_
$ ls out_*
out_aa  out_ab
$ cat out_aa
ligne8 8
ligne8 8
ligne6 6
$ cat out_ab
lOgne6 6
ligne8 8
```

VII.11 Concaténation de lignes : paste

```
paste [ -ds ] fic_1 fic_2 …
```

Le filtre **paste** permet de **concaténer lignes à lignes** les fichiers identifiés par les arguments.

Options

-d' *liste_sép* **'**	:	Définit la liste des séparateurs utilisés pour séparer les lignes des différents fichiers. Le séparateur par défaut est la **tabulation**.
-s	:	Concaténation des fichiers (équivalent à **cat**).

VII.12 Jointure de fichiers : join

```
join [ -options ] fic_1 fic_2
```

Le filtre **join** effectue la **jointure des lignes de deux fichiers triés** sur le champ représentant le critère de jointure. Ce critère est , par défaut, le premier champ de chaque fichier.

Option

-1 ***n***	**:**	Indique le rang du critère, dans le **premier fichier**.
-2 ***m***	**:**	Indique le rang du critère, dans le **second fichier**.
-t *sép*	**:**	Pour définir le séparateur Le séparateur par défaut est le ***blanc***.
-a *fic*	**:**	Pour afficher les lignes d'un fichier ne participant pas à la jointure.
-v *fic*	**:**	Pour n'afficher que les lignes d'un fichier ne participant pas à la jointure.

VIII. SAUVEGARDE et RESTAURATION
La commande tar et le filtre cpio

C'est en général l'administrateur qui est responsable de la sauvegarde (voire restauration) des données d'un serveur Unix.
L'utilisateur doit cependant connaître le fonctionnement de base de ces commandes, de manière à pouvoir être autonome quant à la sauvegarde et restauration de ses propres fichiers.

Il existe deux commandes standards, permettant d'effectuer des sauvegardes et restaurations :
- La commande **tar**.
- Le filtre **cpio**.

Il existe également des commandes permettant de **compresser/décompresser** des fichiers :
- Les commandes **pack/unpack**.
- Les commandes **compress/uncompress**.

VIII.1 ARCHIVAGE DE FICHIERS : La commande tar

La commande **tar** permet :

- De **sauvegarder** une **arborescence de fichiers** sous forme d'archives (format propre à la commande **tar**) grâce à l'option **–c**.
- D'**afficher le contenu d'une sauvegarde** grâce à l'option **–t**.
- De **restaurer** une **arborescence de fichiers** grâce à l'option **–x**.

```
tar -c[v]f archive fichier
tar -t[v]f archive
tar -x[v]f archive fichier
```

Options

-c	:	Pour **sauvegarder** une arborescence de fichiers.
-r	:	Pour **ajouter** des fichiers à une sauvegarde existante.
-u	:	Pour ajouter des fichiers qui n'existent pas ou qui ont été modifiés, à une sauvegarde existante.
-t	:	Pour **afficher** les fichiers d'une sauvegarde.
-x	:	Pour **restaurer** une arborescence de fichiers.
-f	:	Pour **identifier** l'unité ou l'**archive**, utilisée pour la sauvegarde/restauration.
-v	:	Pour afficher la liste des fichiers traités à l'issue de la sauvegarde ou de la restauration.
-z	:	Compacte l'archive avec **gzip**.

Arguments

fichier	**:**	Identifie l'arborescence de **fichiers à sauvegarder ou à restaurer**. Si les fichiers sauvegardés sont identifiés par un **nom relatif**, ils pourront être restaurés n'importe où. Si les fichiers sauvegardés sont identifiés par un **nom absolu**, ils devront être restaurés dans leur emplacement d'origine.

Exemples

```
$ ls -l
total 24
drwxr-xr-x 2 astage15 stage 4096 oct.  28 15:28 backup
drwxr-xr-x 2 astage15 stage 4096 oct.  27 12:05 bin
drwxr-xr-x 2 astage15 stage 4096 oct.  27 12:07 src
-rw-r--r-- 1 astage15 stage   40 oct.  28 10:27 un_autre_fichier
-rw-r--r-- 1 astage15 stage   40 oct.  28 10:24 un_fichier
-rwxr--r-- 1 astage15 stage    0 oct.  27 12:15 un_script

$ tar -cvf backup/sav_`date "+%Y%m%d"`.tar un_*
un_autre_fichier
un_fichier
un_script

$ tar -tvf backup/sav_20141028.tar
-rw-r--r-- astage15/stage   40 2014-10-28 10:27 un_autre_fichier
-rw-r--r-- astage15/stage   40 2014-10-28 10:24 un_fichier
-rwxr--r-- astage15/stage    0 2014-10-27 12:15 un_script

$ rm -v un_fichier
«un_autre_fichier» supprimé
«un_fichier» supprimé

$ tar --wildcards -xvf backup/sav_20141028.tar un_*fichier
un_autre_fichier
un_fichier
```

VIII.2 SAUVEGARDE DE FICHIERS : La commande cpio

La commande **cpio** permet :

- De **sauvegarder des fichiers** (option **–o**) dont les noms sont lus sur le **canal 0**.
  Généralement le canal d'entrée est **redirigé** vers un **fichier d'entrée** ou **connecté** à la sortie d'un '**pipe**'.
  Par défaut, les fichiers sauvegardés sont écrit sur le **canal 1**. Celui-ci doit donc être **redirigé** vers un **fichier de sortie**.
- D'**afficher le contenu d'une sauvegarde** grâce à l'ensemble d'options **–it**.
- De **restaurer des fichiers** (option **–i**) dont l'emplacement de sauvegarde est lu sur le **canal 0**.
  Par défaut, l'ensemble des fichiers sauvegardés, est restauré.

```
cpio –o[options] [ < fic_entrée ] [ > fic_sortie ]
cmde | cpio –o[options] [ > fic_sauveg ]
cpio –it < fic_entrée
cpio –i[options] [ fichiers_a_restaurer ] < fic_sauveg
```

Options

-v	:	Pour afficher la liste des fichiers traités à l'issue de la sauvegarde ou de la restauration.

Exemples

```
$ ls -l
total 24
drwxr-xr-x 2 astage15 stage 4096 oct.  28 15:28 backup
drwxr-xr-x 2 astage15 stage 4096 oct.  27 12:05 bin
drwxr-xr-x 2 astage15 stage 4096 oct.  27 12:07 src
-rw-r--r-- 1 astage15 stage   40 oct.  28 10:27 un_autre_fichier
-rw-r--r-- 1 astage15 stage   40 oct.  28 10:24 un_fichier
-rwxr--r-- 1 astage15 stage    0 oct.  27 12:15 un_script

$ find . -name 'un_*' -type f | cpio -o > backup/sav_`date "+%Y%m%d"`.cpio
1 bloc

$ cpio -it < backup/sav_20141028.cpio
un_autre_fichier
un_fichier
un_script
1 bloc

$ rm -v un_*fichier
«un_autre_fichier» supprimé
«un_fichier» supprimé

$ cpio -i un_*fichier < backup/sav_20141028.cpio
1 bloc
```

IX. L'IMPRESSION

L'impression est l'un des domaines dans lequel les différences entre les systèmes Unix, sont les plus marquées. On distingue en effet trois types de services d'impression :

- Le service d'impression **BSD**, utilisé par les unix **SUN** (**solaris**), **DIGITAL** et **LINUX**.
- Le service d'impression **SYSTEM V**, utilisé par les unix **HP** (**hp ux**) et **SCO**.
- Le service d'impression **AIX** (service propriétaire de l'unix **IBM**).
  (*sujet non traité dans cette brochure)*

Ceci dit, il existe un ensemble de caractéristiques communes à tous les services d'impression.

IX.1 Composants d'un service d'impression

Un service d'impression est composé des éléments suivants :

- Une **commande de lancement d'impression**, grâce à laquelle un utilisateur émet une demande d'impression d'un ou de plusieurs fichiers.
- Des **files d'attente** permettant de stocker les fichiers en attente d'impression.
- Des **imprimantes**, dont chacune peut être associée à une ou plusieurs files d'attente.
- Des **tâches** (les 'daemon' **lpsched** ou **lpd**) permettant d'imprimer les fichiers d'une file d'attente sur une imprimante.
- Des **commandes** permettant d'administrer l'ensemble des composants du service d'impression.

IX.2 Les commandes d'impression

IX.2.1 Systèmes BSD

```
lpr [ -options ] fichier …
commande | lpr [ -options ]
```

Cette commande permet de placer le(s) fichier(s) à imprimer dans la file d'attente associée à l'imprimante par défaut de l'utilisateur (identifiée par la variable d'environnement **PRINTER**).

Options

-P *imprim*	**:**	Pour utiliser une autre imprimante.
-r	**:**	Pour supprimer les fichiers après impression.
-m	**:**	Pour avertir l'utilisateur de la fin de l'impression.
-#n	**:**	Pour imprimer ***n*** exemplaires.

IX.2.2 Systèmes SYSTEM V

```
lp [ -options ] fichier …
commande | lp [ -options ]
```

Cette commande permet de placer le(s) fichier(s) à imprimer dans la file d'attente associée à l'imprimante par défaut de l'utilisateur (identifiée par la variable d'environnement **PRINTER**).

Options

-d *imprim*	**:**	Pour utiliser une autre imprimante.
-m	**:**	Pour avertir l'utilisateur de la fin de l'impression.
-n	**:**	Pour imprimer **n** exemplaires.
-t *titre*	**:**	Pour imprimer une 'bannière' comportant le titre spécifié.
-q *priorité*	**:**	Pour affecter une priorité (**0-haute**, **39-basse**) à la demande d'impression.

IX.3 Les commandes de gestion des files d'attente et des imprimantes

IX.3.1 Systèmes BSD

Affichage des files d'attente

```
lpq [ -options ]
```

Cette commande permet d'afficher la liste des demandes d'impression stockéesdans les files d'attente associée à l'imprimante par défaut de l'utilisateur (identifiée par la variable d'environnement **PRINTER**). Chaque demande d'impression est associée à un **numéro de job**.

Options

-P *imprim*	**:**	Pour afficher les files d'attente d'une autre imprimante.

Affichage d'informations sur les imprimantes disponibles

```
lpc status
```

Les informations affichées indiquent pour chaque imprimante, si l'imprimante est en service, et si sa file d'attente est active.
La plupart des options de cette commande sont réservées à l'utilisateur root.

IX.3.2 Systèmes SYSTEM V

```
lpstat [ -options ]
```

Cette commande permet d'afficher des informations relatives au service d'impression :
- Liste et état des imprimantes disponibles.
- Liste et contenu des files d'attente. Chaque demande d'impression est associée à un identifiant constitué du nom de l'imprimante, d'un tiret et d'un numéro.

Par défaut ces informations ne concernent que l'utilisateur en cours.

Options

-a *imprim*	**:**	Pour afficher l'état des imprimantes spécifiées.
-u *compte*	**:**	Pour afficher les informations concernant les utilisateurs spécifiés.
-d	**:**	Pour afficher le nom de l'imprimante par défaut.
-t	**:**	Pour afficher toutes les informations.

IX.4 Annulation de demandes d'impression

IX.4.1 Systèmes BSD

```
lprm [ -options ] [ numéro_job ]
```

Cette commande **supprime la demande d'impression** correspondant au numéro de job spécifié.
Par défaut, la dernière demande d'impression est supprimée.
Si l'argument passé est un – (au lieu d'un numéro de job), l'ensemble des demandes de l'utilisateur sont supprimées.

Options

-P *imprim*	**:**	Pour utiliser les files d'attente d'une autre imprimante.

IX.4.2 Systèmes SYSTEM V

```
cancel numéro_requête
cancel imprimante
```

Cette commande **supprime** :
- La **demande d'impression** correspondant au **numéro spécifié**.
- L'**impression** en cours sur l'**imprimante spécifiée**.

X. Principes de base de la SECURITE UNIX

X.1 Le COMPTE UTILISATEUR

Les comptes utilisateur sont utilisés :
- Pour se **connecter au système**.
- Pour identifier l'utilisateur qui est **propriétaire** d'un fichier.

```
root:x:0:0:root:/root:/bin/bash
astage15:x:1015:500::/home/stage/astage15:/bin/bash
alain:x:1061:1061::/home/alain:/bin/bash
```

Les caractéristiques d'un compte sont stockées dans le fichier **/etc/passwd**. Ce sont :
- Le **nom de connexion** (affiché par la commande **logname**).
- Le **mot de passe**.
- L'**identifiant utilisateur** (**UID** : **U**ser **ID,** affiché par la commande **id**)
- L'**identifiant** de **groupe primaire** (**GID** : **G**roup **ID,** affiché par la commande **id**)
- Un **commentaire** (contenu libre) qui peut être utilisé par certaine commande, notamment la commande **finger**.
- Le **répertoire de connexion** qui permet d'identifier le **répertoire en cours** positionné à la connexion de l'utilisateur. Son chemin d'accès absolu est mémorisé dans la variable d'environnement **$HOME**.
- Le **programme de connexion** qui exécuté à la connexion. Il s'agit généralement d'un **shell**. Son chemin d'accès absolu est mémorisé dans la variable d'environnement **$SHELL**
  L'utilisateur est déconnecté dès la fin de ce programme.

X.2 Les GROUPES

Un compte utilisateur est obligatoirement associé à un groupe primaire. Le compte peut cependant être invités par d'autres groupes, qui sont ses groupes secondaires.

```
stage:x:500:
```

Il convient de distinguer l'importance et le rôle de ces 2 types de groupes :
- Lorsqu'un utilisateur créé un fichier, celui-ci est associé au **groupe primaire** du compte propriétaire, permettant aux autres membres du groupe de bénéficier de **droits**.
- L'appartenance à un **groupe secondaire** permet d'accéder aux fichiers associés à ce groupe, sans avoir à changer son groupe primaire.

Les caractéristiques d'un groupe sont stockées dans le fichier **/etc/group**. Ce sont :
- Le **nom du groupe**.
- Un champ vide, correspondant au mot de passe du groupe (**obsolète**).
- L'**identifiant** du **groupe** (**GID** : **G**roup **ID**)
- La **liste des comptes membres de ce groupe**.

X.3 Commandes liées à la gestion des comptes et des groupes

X.3.1 Changement de mot de passe

```
passwd [ compte ]
```

Cette commande permet à un utilisateur de changer son mot de passe. Son mot de passe actuel lui sera demandé, ainsi que le nouveau mot de passe et une confirmation du nouveau mot de passe.
Seul, l'utilisateur root est autorisé à passer un nom de compte en argument.

X.3.2 Modification du groupe primaire d'un processus

```
newgrp id_groupe
```

Cette commande génère un nouveau **processus shell,** associé au groupe dont le nom est passé en argument. Elle permet à l'utilisateur de créer des fichiers qui seront associés à un autre groupe que son groupe primaire habituel.
L'utilisateur doit être 'invité' du groupe spécifié.
Le retour à la situation initiale se fait par **exit**.

X.3.3 Modification du groupe primaire d'un fichier

```
chgrp nom_groupe fichier(s)
```

Cette commande permet modifier le groupe primaire d'un ou de plusieurs fichiers.

X.3.4 Modification du propriétaire d'un fichier

```
chown nom_compte fichier(s)
```

Cette commande permet modifier le propriétaire d'un ou de plusieurs fichiers.

X.3.5 Adoption des droits d'un autre utilisateur

```
su [ - ] nom_compte
```

Cette commande génère un nouveau **processus shell,** disposant des privilèges de l'utilisateur dont le nom est passé en argument.

- **Sans tiret**, les fichiers d'initialisation de l'utisateur spécifié ne sont pas exécutés.
- **Avec tiret**, les fichiers d'initialisation de l'utisateur spécifié sont exécutés.

X.4 Mode d'un fichier

Unix peut considèrer l'utilisateur qui accède à un répertoire ou à un fichier de 3 façons différentes.
- Soit l'utilisateur est le **propriétaire** du répertoire ou du fichier (**owner**).
- Soit l'utilisateur est membre du **même groupe primaire** que celui du **propriétaire** (**group**).
- Soit l'utilisateur est ni l'un, ni l'autre (**others**).

Chaque catégorie d'utilisateur peut disposer de 3 droits, manipulés à l'aide de 2 types de notations :
la **notation symbolique** et la **notation octale**.:
- Droit de **lecture** (**r** ou .**4**, situé à gauche d'un groupe de droits).
- Droit d'**écriture**. (**w** ou .**2**, situé au centre d'un groupe de droits).
- Droit d'**exécution**. (**x** ou .**1**, situé à droite d'un groupe de droits).

Ces droits sont stockés dans l'**inode** du fichier, et peuvent être affiché par **ls –l**.
Ils constitue, avec le type de fichier, ce que l'on appelle le **mode du fichier**.

	owner	group	others
Notation **symbolique**	rwx	r-x	r-x
Notation **binaire**	111	101	101
Notation **octale**	7	5	5

```
$ mkdir mon_rep
$ touch mon_fichier
$ ls -ld mon_*
-rw-r--r-- 1 astage15 stage    0 oct.  29 10:56 mon_fichier
drwxr-xr-x 2 astage15 stage 4096 oct.  29 10:56 mon_rep
```

X.4.1 Evaluation des droits par défaut : La commande umask

```
umask [ nouveau_mask ]
```

Lors de sa création, un fichier ou un répertoire acquiert des **droits par défaut**, évalués en fonction de la valeur renvoyée par la commande **umask** (par défaut **022**).
Ces droits sont évalués de la façon suivante :
- Pour les **répertoires** : **777 - umask**
- Pour les **fichiers** : **666 - umask**

X.4.2 Droits d'accès aux fichiers et aux répertoires

LECTURE : **r**
- **Répertoire** : Le contenu du répertoire est visualisable (**ls** par exemple).
- **Fichier** : Le contenu du fichier est visualisable (**cat** par exemple).

ECRITURE : **w**
- **Répertoire** : Le contenu du répertoire est modifiable (création, suppression de fichiers…)
- **Fichier** : Le contenu du fichier est modifiable (**vi** par exemple).

EXECUTION : **x**
- **Répertoire** : Indispensable pour pouvoir **accéder au contenu du répertoire**.
- **Fichier** : Autorise l'exécution du fichier en saisissant simplement son nom.

X.5 Modification des droits d'un fichier : la commande chmod

La commande **chmod** permet au propriétaire d'un fichier ou à l'administrateur, de modifier les droits accordés sur ce fichier.

```
chmod nouveau_droits fichier(s)
```

Les nouveaux droits peuvent être spécifiés en utilisant la **notation symbolique** ou la **notation octale**.

Notation symbolique

Cette méthode consiste à indiquer qui est concerné par l'opération, la nature de l'opération et le type de droit manipulé.

Qui		
• u	:	Pour le **propriétaire**
• g	:	Pour le **groupe**
• o	:	Pour les **autres**
• a	:	Pour les 3 types d'utilisateurs

Opération		
• +	:	Pour **ajouter** les droits spécifiés
• -	:	Pour **supprimer** les droits spécifiés
• =	:	Pour **remplacer** les droits existants par les droits spécifiés

Droit		
• r	:	Pour le droit de **lecture**
• w	:	Pour le droit d'**écriture**
• x	:	Pour le droit d'**exécution**

Exemples

```
$ ls -ld mon_*
-rw-r--r-- 1 astage15 stage    0 oct.  29 10:56 mon_fichier
drwxr-xr-x 2 astage15 stage 4096 oct.  29 10:56 mon_rep

$ chmod ug+x mon_fichier
$ chmod g+w,o-rx mon_rep
$ ls -ld mon_*
-rwxr-xr-- 1 astage15 stage    0 oct.  29 10:56 mon_fichier
drwxrwx--- 2 astage15 stage 4096 oct.  29 10:56 mon_rep
```

X.6 Les droits étendus

Il existe trois droits supplémentaires : le **Suid**, le **Sgid** et le **Sticky Bit**.

X.6.1 Droit SUID

- **Fichier** : Permet d'**exécuter le fichier en adoptant les droits du propriétaire** du fichier.
  C'est le cas de certaines commandes, **passwd** par exemple.
  On accorde ce droit en spécifiant :
  - **u+s** avec la notation symbolique.
  - **4nnn** avec la notation octale.
- **Répertoire** : Aucun intérêt.

```
$ ls -l /usr/bin/passwd
-r-s--x--x    1 root     root        16336 fév 13  2003 /usr/bin/passwd
Le "s" minuscule indique, en plus du droit SUID, la présence du droit d'exécution, tandis
qu'un "S" majuscule indiquerait son absence.
```

X.6.2 Droit SGID

- **Fichier** : Permet d'**exécuter le fichier en adoptant les droits du groupe primaire** dont le propriétaire est membre. On accorde ce droit en spécifiant :
  - **g+s** avec la notation symbolique.
  - **2nnn** avec la notation octale.
- **Répertoire** : Permet, lors de la création d'un fichier, d'associer ce dernier au **groupe primaire du propriétaire du répertoire**.

X.6.3 Le Sticky Bit

- **Fichier** : Rend le programme exécutable **résident** en mémoire (désormais peu utilisé).
- **Répertoire** : Indique qu'un fichier de ce répertoire ne peut être détruit que par son propriétaire.
  On accorde ce droit en spécifiant :
  - **u+t** avec la notation symbolique.
  - **1nnn** avec la notation octale.

XI. LES PROCESSUS

XI.1 Notion de processus

On appelle **processus** tout programme (ou commande) en cours d'exécution. Chaque processus réclame une certaine quantité de ressources :

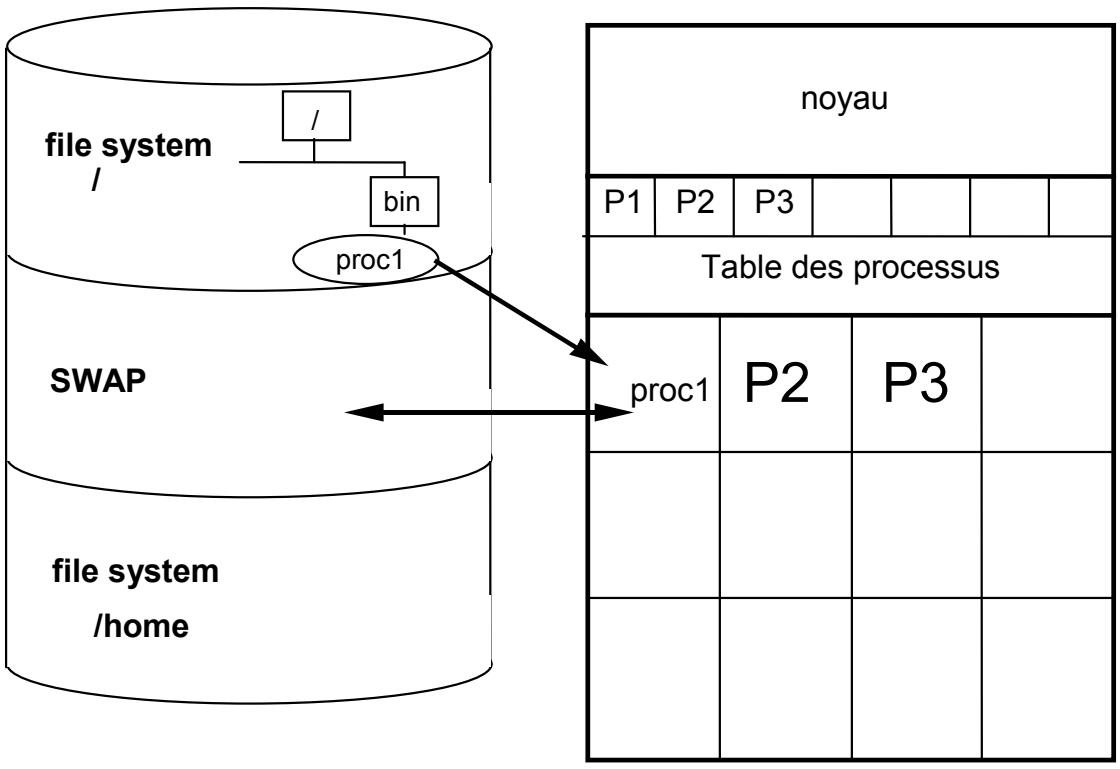
- Processeur
- Mémoire (**ram**)
- Disque (**partition de swap**)

Tout processus est engendré par un autre processus. On parlera ainsi de **processus fils** et de **processus père**.

- Si le **processus fils** s'exécute en avant-plan (**foreground**), son père se met en attente, et ne reprend son exécution qu'après avoir reçu le **code-retour** de son fils.
- Si le **processus fils** s'exécute en arrière-plan (**background**), son père reprend la main immédiatement.

Le système conserve en mémoire la trace et les caractéristiques de tous les processus grâce à la **table des processus**. C'est cette table que l'on consulte lorsque l'on exécute la commande **ps**.
Parmi les principales caractéristiques d'un processus, on distingue :

- Son **pid** (**P**rocess **ID**) permettant de l'identifier de manière unique.
- Son **ppid** (**P**arent **P**rocess **ID**) permettant de l'identifier de manière unique.
- Son **RUID** (**R**eal **U**ser **ID**) identifiant l'utilisateur qui l'a exécuté.
- Son **EUID** (**E**ffective **U**ser **ID**) identifiant l'utilisateur dont on adopte les droits, en cas de droit **SUID**.



XI.2 Gestion de la mémoire et états de processus

Dès la demande d'exécution d'un programme (ou d'une commande), le processus est créé. Des pages sont alors transférées de la partition dans laquelle le fichier est stocké vers la mémoire vive, au fur et à mesure du traitement.
Par la suite, certaines (voire toutes) pages actives peuvent être transférées dans la **partition de swap**, lorsqu'elles ne sont pas utilisées et que l'espace libre en mémoire vive devient insuffisant.

En cours d'exécution, un processus peut connaître **différents états**, dont les principaux sont les suivants.

- **Actif**
  Les pages nécessaires à l'exécution du processus sont en mémoire vive, et le processus peut prétendre à l'utilisation du processeur.
- **En cours d'exécution**
  Les pages nécessaires à l'exécution du processus sont en mémoire vive, et le processus détient le contrôle du processeur.
- **Attente**
  Le processus est en attente d'une E/S (physique ou saisie clavier, par exemple). Certaines pages du processus sont éventuellement transférées dans la **partition de swap**.
- **Stoppé** (ou **suspendu**)
  Le processus est temporairement arrêté (**ctrl<Z>**). Ses pages sont transférées dans la **partition de swap**.
- **zombie**
  Le processus, bien que terminé, est toujours référencé dans la table des processus actifs. Il restera dans cet état (**mort-vivant**) tant que le processus père n'est pas informé de son code-retour.

XI.3 Environnement d'un processus

Lorsqu'un processus s'exécute dans son propre environnement, celui-ci est principalement constitué de trois régions.

- Son **code exécutable**.
- Son **contexte local**, contenant ses données sous forme de **variables locales**.
- Son **contexte global**, contenant la table de descripteurs des fichiers ouverts, le **UID** et **GID** et les **variables globales** (ou **variables d'environnement**).
  Ce contexte est en fait une copie de celui de son père.

Dans certains cas, un processus peut s'exécuter dans l'environnement de son père.

XI.4 Arrêt d'un processus et retour au père

```
exit [ code_retour ]
```

Cette commande permet de quitter un processus, et de rendre la main au processus père, en lui retournant un code-retour (par défaut : **0**)

XI.5 Affichage des processus actifs : la commande ps

```
ps [ -options ]
```

Exécutée sans option, cette commande affiche certaines caractéristiques des processus fils du processus dans lequel elle est exécutée

- Le **pid** du processus.
- Le nom du fichier spécial gérant le terminal auquel le processus est associé (**tty**).
- L'état du processus (**R** pour prêt, **T** pour stoppé, **Z** pour zombie etc…).
- Son temps cumulé d'exécution.
- La commande (ou programme) en cours d'exécution.

Principales options

La commande présente de nombreuses autres options, que celles présentées ici.

-f	**:**	Pour afficher **toutes les informations** pour chaque processus.
-e	**:**	Pour afficher **tous les processus**.
-a	**:**	Pour afficher les processus les plus fréquents.
-t	**:**	Pour afficher les processus associés aux terminaux spécifiés.

Exemples

```
$ ps -f
UID          PID  PPID  C STIME TTY          TIME CMD
astage15    4883  4882  0 10:42 pts/0    00:00:00 -bash
astage15    4969  4883  0 11:18 pts/0    00:00:00 ps -f
```

XI.6 Gestion des jobs

XI.6.1 Exécution d'un processus en arrière-plan

```
[ nohup ] commande &
```

Le métacaractère **&** permet d'**exécuter un processus en arrière-plan**. 2 informations sont affichées :
- Le numéro de **job** (numéro séquentiel indiquant le rang d'exécution au sein du processus père).
- Le **pid**.

Les périphériques associés aux canaux d'entrée et sortie standards, sont communs aux 2 processus (père et fils). Il est donc parfois nécessaire de rediriger les canaux d'E/S du fils, afin d'éviter les interférences.
La commande **nohup** permet de protéger le processus d'arrière-plan, en cas d'arrêt du processus père.
La commande **jobs** permet d'afficher les caractéristiques des processus d'arrière-plan :
- Le **numéro** du job, suivi d'un + s'il s'agit du plus récent ou d'un – pour le précédent le plus récent.
- L'**état** du job.
- La **commande** en cours d'exécution.

XI.6.2 Changement d'état d'un job

Les étapes suivantes permettent de faire **passer un job d'avant-plan en arrière-plan**.

```
bg %id_job (<ctrl>Z (équivalent à kill -STOP))
```

Les étapes suivantes permettent de faire **passer un job d'arrière-plan en avant-plan**.

```
fg %id_job
```

Identification du job

%*n*	:	Pour identifier le ***nième*** job.
%%	:	Pour identifier le **dernier** job.
%-	:	Pour identifier l'**avant-dernier** job.
%*chaine*	:	Pour identifier le job dont la commande commence par la chaîne spécifiée.
%?*chaine*	:	Pour identifier le job dont la commande contient la chaîne spécifiée.

Exemples

```
$ sleep 3600
[2]+  Stopped                 sleep 3600
$ bg %2
[2]+ sleep 3600 &
$ jobs
[1]-  Running                 xterm &
[2]+  Running                 sleep 3600 &
$ kill -15 %2
$ jobs
[1]-  Running                 xterm &
[2]+  Complété                sleep 3600
```

XI.6.3 Gestion des signaux : les commandes kill et trap

La commande **kill** permet à un processus (émetteur) d'envoyer un signal à un autre processus (récepteur).
La commande **trap** permet à un processus (récepteur) d'intercepter et de traiter un signal.

```
kill –l
kill [ –signal ] pid
trap 'commande' signal
```

L'option **–l** de la commande **kill** donne la liste de signaux disponibles.

Principaux signaux

2 ou **INT**	:	Arrêt brutal d'un processus (équivalent à **<ctrl>C**).
9 ou **KILL**	:	Arrêt brutal d'un processus, sans possibilité d'interception.
15 *ou* **TERM**	:	Arrêt normal d'un processus (équivalent à **exit**).
19 ou **STOP**	:	Suspension d'un processus (équivalent à **<ctrl>Z**).
18 ou **CONT**	:	Reprise d'exécution d'un processus suspendu.

XII. L'éditeur VI

XII.1 Appel et présentation de vi

```
vi [ -options ] [ fichier … ]
```

Options

-r	: Permet la reprise du fichier après une interruption système
+n	: Affichage du fichier au numéro de ligne indiqué

L'éditeur propose **2 modes** de fonctionnement :

- Le **mode commande** dans lequel les touches du clavier correspondent à des commandes.
- Le **mode insertion** qui autorise la saisie de texte.

Le **passage du mode saisie au mode commande** se fait par la touche **échappement** (**esc**).

XII.2 Passage au mode INSERTION

a	:	Insertion de texte après le curseur
A	:	Insertion de texte en fin de ligne
i	:	Insertion de texte devant le curseur
I	:	Insertion de texte en début de ligne
o	:	Insertion de lignes sous le curseur
O	:	Insertion de lignes au-dessus du curseur

XII.3 Commandes de déplacement dans le texte

k	**:**	Ligne précédente
j	**:**	Ligne suivante
l	**:**	Déplacement à droite
h	**:**	Déplacement à gauche
***n*+**	**:**	Positionnement sur le premier caractère de la nième ligne suivante
***n*-**	**:**	Positionnement sur le premier caractère de la nième ligne précédente
0	**:**	Curseur en début de ligne
$	**:**	Curseur en fin de ligne
G	**:**	Aller en fin de fichier
***n*G**	**:**	Aller en ligne n
***n*w**	**:**	Curseur sur le nième mot suivant
***n*b**	**:**	Curseur sur le nième mot précédent
***n*e**	**:**	Curseur à la fin du nième mot
H	**:**	Curseur sur la première ligne de l'écran
L	**:**	Curseur sur la dernière ligne de l'écran
ctrl F	**:**	Page suivante
ctrl B	**:**	Page précédente
ctrl D	**:**	Demi-écran suivant
ctrl U	**:**	Demi-écran précédent
ctrl E	**:**	Déplacement d'une ligne vers l'avant
ctrl Y	**:**	Déplacement d'une ligne vers l'arrière

Commandes de déplacement dans le texte (suite)

z <*rc*>	**:**	Ligne courante en haut de l'écran
z.	**:**	Ligne courante au centre de l'écran
m*x*	**:**	Marque la position courante avec un caractère (a à z) x: déplace le curseur sur la position marquée

XII.4 Commandes de correction de texte

R	**:**	Modifie le texte à partir du curseur
s	**:**	Supprime le caractère à partir du curseur et passe en mode insertion
S	**:**	Modifie la ligne
C	**:**	Modifie la ligne à partir de la position du curseur
r*x*	**:**	change le caractère sous le curseur par le caractère précisé (*x*)
***n*x**	**:**	Supprime n caractères à partir du curseur
***n*X**	**:**	Supprime n caractères avant le curseur
***n*dd**	**:**	Supprime n lignes à partir du curseur**.** Les lignes supprimées sont bufférisées
***n*dw**	**:**	Supprime *n* mots à partir du curseur. Les mots sont bufférisés
D	**:**	Supprime à partir du curseur jusqu'à la fin de la ligne et bufférise les caractères supprimés
***n*yy**	**:**	Bufférise n lignes à partir du curseur
***n*yw**	:	Bufférise n mots à partir du curseur
y$	:	Bufférise à partir du curseur jusqu'à la fin de la ligne
p	:	Restaure le contenu du buffer après le curseur
***n*"xyy**	**:**	Copie n lignes dans le buffer x (a à z)
"xp	:	Restaure le contenu du buffer x
u	**:**	Annule la dernière modification
U	**:**	Restaure la ligne courante
***n*>>**	**:**	Idente *n* lignes vers la droite
***n*<<**	**:**	Idente *n* lignes vers la gauche
J	**:**	Joint la ligne courante à la ligne suivante

XII.5 Commandes "ex" : Multi-édition et sauvegardes

:x	**:**	Sauvegarde le fichier en cours et quitte l'éditeur (équivalent à **ZZ**).
:q	**:**	Quitte l'éditeur (impossible si des modifications ont été apportées et non sauvegardées).
:q!	**:**	Quitte l'éditeur sans prises en compte des modifications.
:w	**:**	Sauvegarde le fichier en cours. Un nouveau de fichier peut être spécifié.
:ln,*lmw*	**:**	Sauvegarde les lignes *n* à m dans le fichier spécifié.
:r *fic*	**:**	Insère le contenu du fichier sous le curseur
:e *fic*	**:**	Remplace le fichier en cours par le fichier spécifié.
:e #	**:**	Rappelle en mémoire le dernier fichier chargé (permet de commuter entre *n* fichiers)
:!*cmde*	**:**	Pour exécuter une commande.
:r!*cmde*	**:**	Pour exécuter une commande et placer le résultat derrière la ligne active.

XII.6 Commandes de recherche et remplacement de texte

Ces commandes autorisent l'utilisation de métacaractères.

/*chaine*	**:**	Recherche d'une chaine de caractères, vers la fin du fichier. **n** pour répéter la derniére recherche.
?*chaine*	**:**	Recherche d'une chaine de caractères, vers le début du fichier. **N** pour répéter la derniére recherche.
:g/*chaine*/**d**	**:**	Détruit toutes les lignes contenant la chaine spécifiée.
:v/*chaine*/**d**	**:**	Détruit toutes les lignes ne contenant pas la chaine spécifiée.
:*x,y***s/***ch1*/ *ch2*/	**:**	Remplace la première occurrence de ***ch1*** par ***ch2***, des lignes ***x*** à ***y*** incluses.
:*x,y***s/***ch1*/ *ch2*/**g**	**:**	Remplace toutes les occurrences de ***ch1*** par ***ch2***, des lignes ***x*** à ***y*** incluses.

XIII. COMPLEMENTS

XIII.1 Les travaux « batchs »

XIII.1.1 Exécution différée d'une commande : la commande at

La commande **at** permet de différer l'exécution de commande(s) :
- Lue(s) à partir du **canal 0** (entrée standard).
- Lue(s) dans le fichier spécifié par l'option **–f**.

Si le traitement produit un **résultat**, celui-ci est **transmis à l'utilisateur par mail**.

```
at [ -options ] heure [ date ] [ +incrément ]
```

Options

-f	:	Pour identifier un fichier contenant des commandes dont l'exécution doit être différée.
-m	:	Pour envoyer un **mail** avertissant l'utilisateur de la fin d'exécution de la commande.
-l	:	Pour afficher la liste des jobs en attente d'exécution.
-r	:	Pour supprimer le job dont le numéro est spécifié.

Arguments

heure	:	L'heure d'exécution doit être indiquée sosu la forme ***HHMM*** ou ***HH:MM.*** Elle peut également être spécifiée à l'aide de valeurs spéciales, impliquant également l'utilisation d'un incrément : **now** pour exprimer l'heure à partir de l'heure en cours. **midnight** pour exprimer l'heure à partir de minuit (00:00). **noon** pour exprimer l'heure à partir de midi (12:00).
date	:	Une date peut être spécifiée sous l'une des formes ***MMJJAA***, ***MM/JJ/AA*** ou ***MM.JJ.AA***.
incrément	:	Si l'heure est spécifiée de manière relative, un incrément doit fourni en utilisant l'une des valeurs **minutes, hours, days, weeks, months, years**.

```
$ at 11:53
warning: commands will be executed using /bin/sh
at> echo "essai at" > ~astage15/sortie_at
at> <EOT>
job 37 at Wed Oct 29 11:53:00 2014
$ atq
37      Wed Oct 29 11:53:00 2014 a astage15
$ date
mercredi 29 octobre 2014, 11:51:37 (UTC+0100)
$ ls sortie_*
sortie_at
$ atq
$
```

XIII.1.2 Exécution cyclique d'une commande : la commande crontab

Ce type de jobs est gérés par le **démon cron** (ou **crond** sur certains systèmes)
La commande **crontab** soumet au démon, le contenu du fichier dont le nom est spécifié et dont la structure des lignes est imposée.
Chaque utilisateur ne peut disposer que d'un seul de ses fichiers.

```
crontab fichier
```

Structure d'une ligne du fichier

- **Champ 1 :** Indication de **minutes** (0 à 59).
- **Champ 2 :** Indication de **heures** (0 à 23).
- **Champ 3 :** Indication du **jour dans le mois** (1 à 31).
- **Champ 4 :** Indication du **mois** (1 à 12).
- **Champ 5 :** Indication du **jour dans la semaine** (0 à 6, le premier jour étant le Dimanche).

Pour chaque champ, la valeur peut être spécifiée sous l'une des formes suivantes.
- Une **valeur**.
- Une **liste de valeurs**, séparées par des virgules.
- Une **tranche de valeurs** (une valeur minimum et une valeur maximum séparées par un tiret).
- Une combinaison des 2 méthodes précédentes
- Le métacaractère * représentant toutes les valeurs possibles.

Le **champ 6** contient la commande à exécuter.

Exemples

```
$ cat cron_ae
# minutes heure jour mois jour_semaine commande #
  45      11    6    1    4            echo ***Test Cron*** > /dev/pts/0
$ crontab cron_ae
…et soudain…
$ ***Test Cron***
```

XIII.2 Les commandes de messagerie

XIII.2.1 Affichage d'informations internes : la commande news

Dans certains systèmes Unix, cette commande permet aux utilisateurs d'afficher des « bulletins internes », généralement stockés par l'administrateur dans le répertoire **/usr/news**.
Chaque fois qu'un bulletin est lu, le fichier $HOME/.news_stamp est mis à jour, permettant ainsi aux bulletins dont la date est antérieure à celle de ce fichier de ne plus être lus.

XIII.2.2 Envoi d'un message : les commandes write et mesg

La commande **write** permet d'**afficher un message sur le terminal d'un utilisateur**.

```
write utilisateur [ tty ]
```

Le message peut –être :
- Saisi au clavier et lu sur l'entrée standard.
- Lu à travers un « **pipe** ».

L'argument ***tty*** peut être utile, si l'utilisateur est connecté sur plusieurs terminaux.
Le destinataire est averti de l'arrivée du message. Il peut autoriser (**mesg y**) ou pas (**mesg n**) la reception de messages.

XIII.2.3 Utilisation d'une boite aux lettres : la commandes mail

La commande **mail** permet de gérer les messages stockés dans la boite aux lettres d'un utilisateur.
Les fichiers mails (boites aux lettres) sont stockés dans le répertoire **/usr/spool/mail**.

```
mail [ -f fichier ] [ utilisateur ]
```

Le texte du message peut-être :
- Saisi sur le clavier (lecture sur l'entrée standard)
- Lue(s) dans le fichier spécifié par l'option **–f**.

Les différentes sous-commandes disponibles, peuventêtre affichées par la sous-commande **help** ou **?**.