

Module Station de Travail Initiation au système UNIX

Serge Gadret, Philippe Dax

22 août 1997

Table des matières

0.1	INTRODUCTION	3
1	SYSTEME DE FICHIERS	4
1.1	L'ARBORESCENCE	4
1.1.1	Généralités	4
1.1.2	Un système de fichiers	5
1.1.3	Arborescence locale	6
1.1.4	Arborescence réseau	8
1.2	COMMANDES DE BASE	9
1.2.1	Chemin d'accès à un élément de l'arbre	9
1.2.2	Parcours de l'arbre	9
1.2.3	Les répertoires	10
1.2.4	Les fichiers	11
2	LES PROCESSUS	16
2.1	GENERALITES	16
2.1.1	Définition	16
2.1.2	Création	16
2.1.3	Les entrées sorties	17
2.1.4	Communication	18
2.1.5	Contrôle	19
2.1.6	Les signaux	19
2.2	CARACTERISTIQUES	20
2.2.1	Type	20
2.2.2	Identité	21
2.2.3	Etat	21
2.2.4	Mode d'exécution	21
2.3	LA COMMANDE ps	21
3	L'INTERPRETEUR DE COMMANDES	25
3.1	GENERALITES	25
3.2	OUVERTURE D'UNE SESSION	26
3.3	LA LIGNE DE COMMANDE	26
3.3.1	Redirections	28
3.3.2	Enchaînement de commandes	28
3.3.3	Mode d'exécution	29
3.3.4	Méta-caractères	29
3.4	BOURNE SHELL	30
3.4.1	Les variables du sh	30
3.4.2	Fichiers d'initialisation	31
3.4.3	Traitement des signaux	31

3.5	LE ZSH	33
3.5.1	Les variables du zsh	33
3.5.2	Fichiers d'initialisation	33
3.5.3	Alias	33
3.5.4	Historique des commandes	34
3.5.5	Complètement des noms de fichiers	34
3.6	L'ENVIRONNEMENT ENST	34
4	COMMANDES UNIX	37
4.1	COMMANDES ELEMENTAIRES	37
4.2	AUTRES COMMANDES	43
4.3	RÉSUMÉ DES COMMANDES VI	51

0.1 INTRODUCTION

Le système UNIX conçu en 1969 aux Bell-Labs (AT&T), a évolué suivant deux familles principales. L'une développée au sein de ATT et des Bell Laboratories a abouti à la version SYSTEM V, l'autre développée à l'université de Berkeley a donné naissance aux versions BSD (Berkeley System Distribution). Chaque version ayant développé ses propres fonctionnalités.

SYSTEM V : – IPC

- File de messages
- mémoire partagée
- Sémaphores

BSD : – Gestion de mémoire virtuelle

- Communication IPC socket
- Ethernet TCP/IP (ftp, telnet, remote-commands)

Après cette période de divergence, des propositions de standardisation ont vu le jour (POSIX) dans le but de rendre compatibles ces différentes versions d'UNIX et en conservant les spécificités propres à chaque version. Le standard se rapprochant d'avantage de System V.

Les versions commerciales intègrent les fonctionnalités des deux versions. Le parc de stations de travail de l'ENST est équipé principalement de machines SUN, avec son système d'exploitation SUNOS.

Ce système d'exploitation basé à l'origine sur la souche BSD, a évolué dans ses dernières versions vers SYSTEM V, il intègre également les spécificités des deux souches. Il est à noter que SUN a développé ses propres fonctionnalités.

- NFS (Network File System): Permet d'accéder aux systèmes de fichiers des autres machines.
- NIS (Network Informations Services):Gestion centralisée de différents fichiers d'administration.

Chapitre 1

SYSTEME DE FICHIERS

1.1 L'ARBORESCENCE

1.1.1 Généralités

Le système de fichier UNIX est basé sur une structure arborescente hiérarchisée. Le système ne connaît aucune notion d'organisation de fichiers classique (séquentielle, indexée ...). Les fichiers sont constitués d'une chaîne d'octets non structurée du point de vue du système. L'arborescence représentée ci-dessous est composée des éléments suivants:

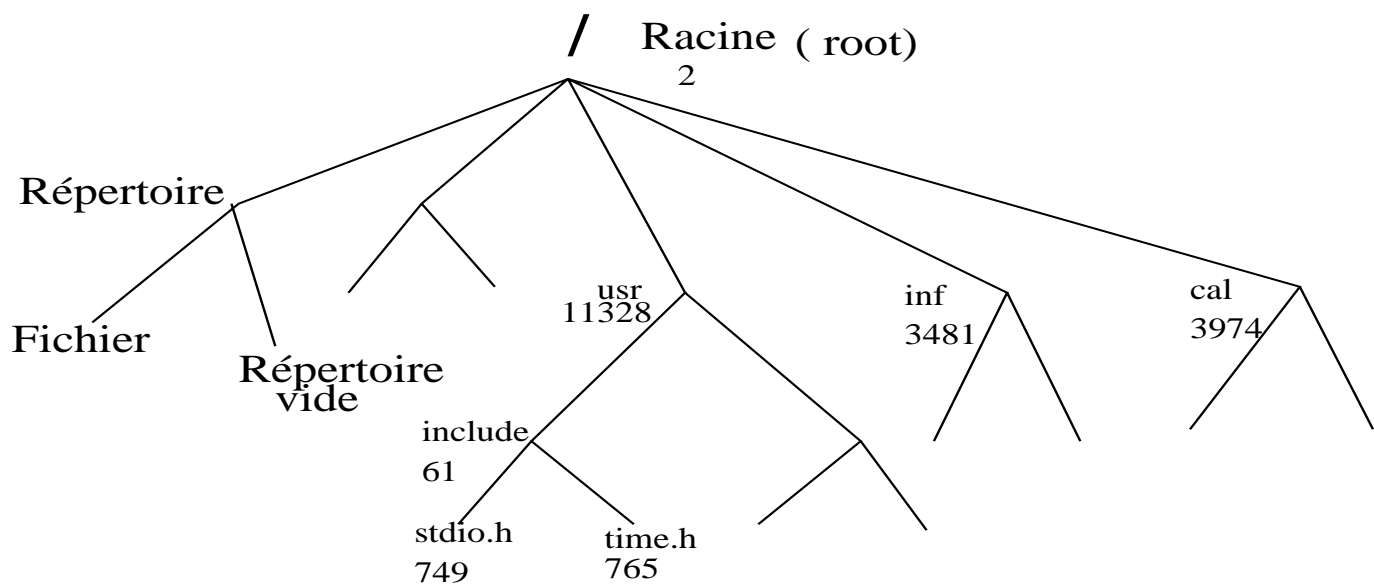


FIG. 1.1 – Arborescence

Sommet de l'arbre: la racine (**root**) sera représentée par slash (/) au début de la chaîne de caractères représentant le chemin d'accès en absolu à un fichier ou répertoire.

- /usr/bin/lis
- /var/spool/mail/dupont

Noeud : Répertoire contenant la liste des fichiers ou sous-répertoires

Feuilles: Fichiers qui sont classés en deux catégories:

- Fichiers **normaux**: Texte, binaire, son, image.
- Fichiers **spéciaux**: Ils se divisent en deux catégories suivant le mode d'accès.
 - Mode **bloc**: disques, disquettes.
 - Mode **caractères**: Terminaux, lecteurs de bande.

Chaque élément de l'arbre, répertoire ou fichier, sera identifié par un nom: référence pour l'utilisateur. A ce nom sera associé un numéro, sous forme d'entier (numéro d' **inode**) qui sera la référence pour le système. On notera d'autre part que les périphériques sous UNIX sont banalisés en étant vus comme des fichiers pour l'utilisateur.

1.1.2 Un système de fichiers

L'arbre représentant l'organisation des fichiers et répertoires, est composé de plusieurs systèmes de fichiers. Un système de fichiers est une structure d'accueil sur disque permettant de stocker et de référencer les répertoires et fichiers.

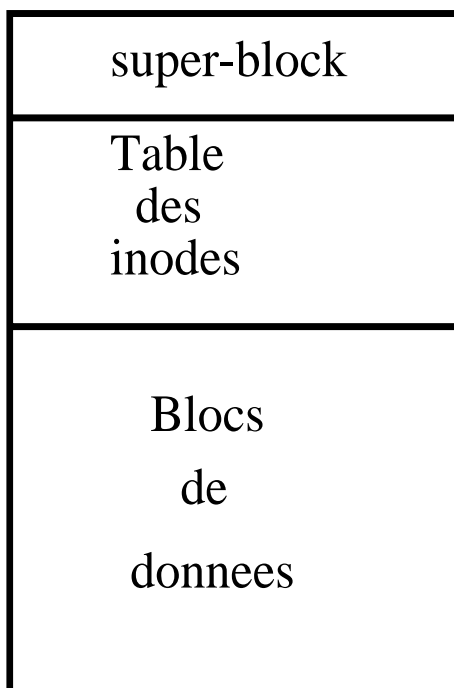


FIG. 1.2 – *Système de fichiers*

Cette structure est composée des éléments suivants:

Super-block: Donne les caractéristiques du système de fichiers: Taille, Tables des blocs libres, nombre de fichiers qu'il peut stocker

Tables des inodes (I-list, index liste): Chaque entrée de la table, correspondant à un numéro d'inode identifiant un fichier ou répertoire, décrit les caractéristiques de ce dernier :

- Taille
- Type (ordinaire, répertoire, spécial).
- Compteur de référence.

- Dates de création/modification, dernier accès.
- Propriétaire, droits d'accès.
- Liste d'adresse des blocs disque où sont stockées les données.

Blocs de données : Partie du disque où est effectivement rangé le contenu des fichiers et répertoires.

1.1.3 Arborescence locale

La Fig 1.3 représente l'arborescence locale d'une machine. Cette machine est équipée d'un disque dur. Ce disque est divisé en disques logiques ou partitions. Chaque partition correspond à un système de fichiers qui a été défini précédemment. Les systèmes de fichiers standards sont :

- `/` et `/usr` : qui supportent le système d'exploitation.
- `/home` : Partition réservée aux données.

Remarque: Sur la figure apparaît un système de fichier swap. Il s'agit d'un système de fichiers spécial utilisé comme mémoire additionnelle à la mémoire vive de la machine.

Voici quelques répertoires à connaître :

`/bin` et `/usr/bin` : contiennent les commandes Unix standard.

`/usr/local/bin` : les commandes supplémentaires issues du domaine public ou développées localement.

`/opt` : les commandes supplémentaires et logiciels tel que *C++*, *fortran*, *pascal*, *Eiffel*, *lisp*, *concerto* ...

`/usr/lib` : contient les bibliothèques de fonctions utilisables par les programmeurs.

`/usr/include` : contient des prédéfinitions d'objets utilisables également en programmation (headers).

`/dev` : contient les fichiers spéciaux associés aux périphériques : disques, lecteurs de bandes, terminaux ...

`/etc` : contient les fichiers nécessaires à la gestion du système. Ces fichiers sont accessibles au simple utilisateur en lecture seulement.

`/tmp` : contient les fichiers temporaires.

`/var` : contient des fichiers journaux, des fichiers en attente de sortie sur imprimante, les boîtes aux lettres du courrier électronique, les news,

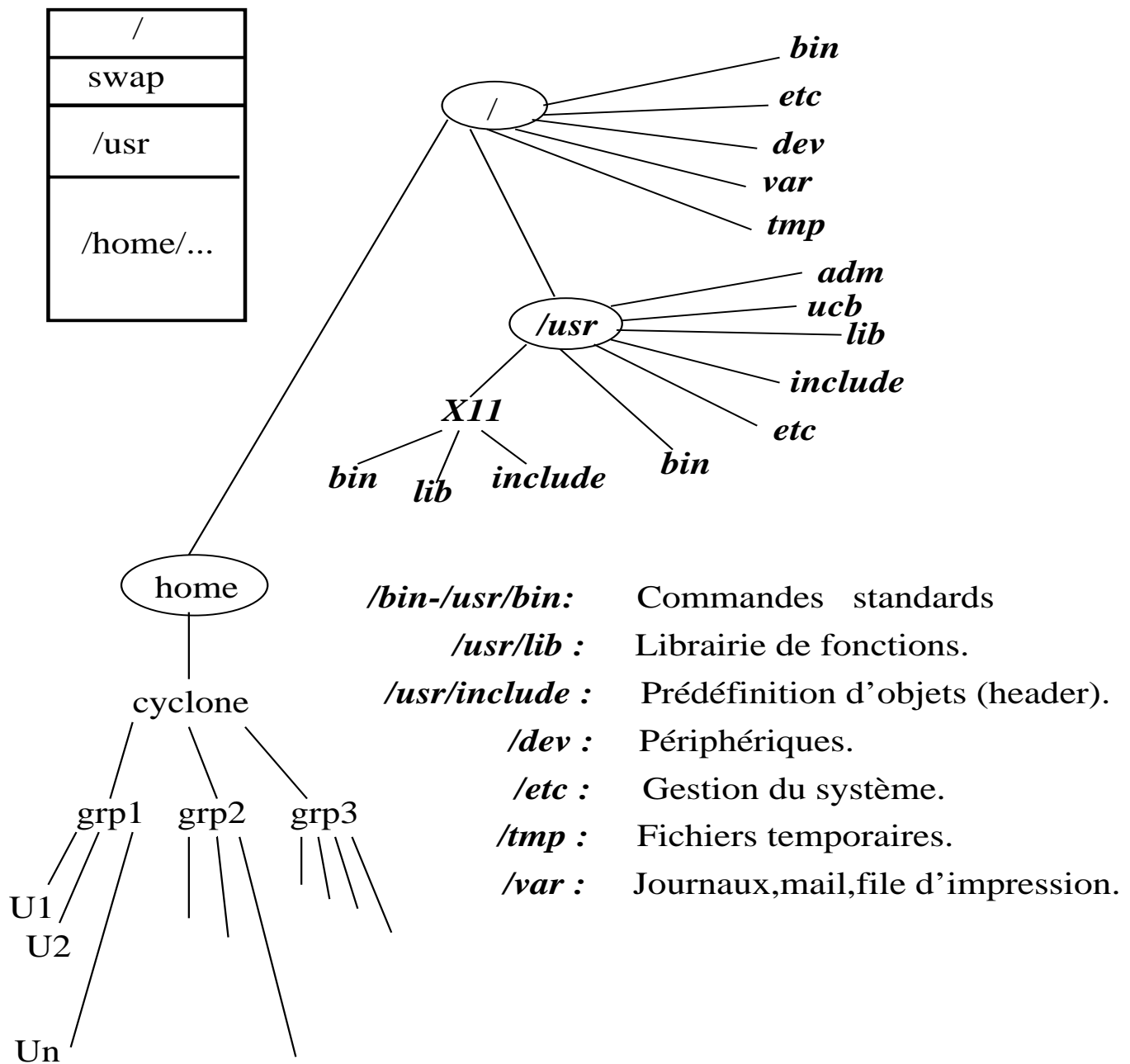


FIG. 1.3 – Arborescence locale

1.1.4 Arborescence réseau

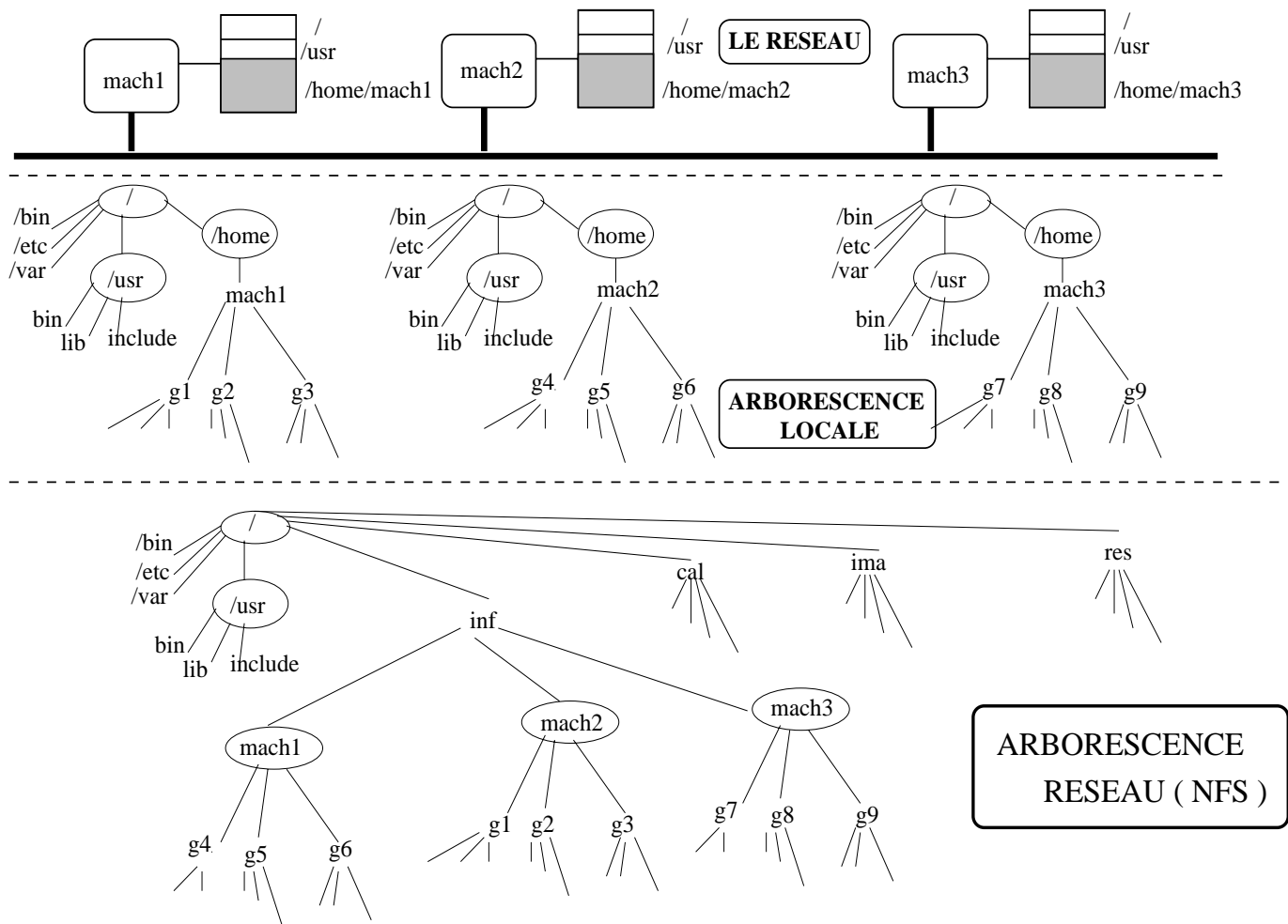


FIG. 1.4 – Arborescence réseau

En se référant à la Fig 1.4, on a trois machines connectées au réseau Ethernet. Pour chaque machine, équipée d'un disque dur, on retrouve l'arborescence locale décrite ci-dessus, c'est à dire comportant les trois systèmes de fichier standards (`/`, `/usr`, `/home/mach?`). Grâce à l'utilisation de NFS (Network File System) développé par SUN, on peut avoir accès depuis une machine à tous les disques des autres machines, sous réserve que celles-ci autorisent le montage de ses disques, et cela de façon entièrement transparente pour l'utilisateur.

On notera que la structure de l'arbre d'une machine quelconque se rapproche de l'organisation physique du réseau. Sous la racine désignée par "`/`" on retrouve des répertoires qui correspondent aux départements de l'école (INF, CAL, SIG, IMA, RES, COM, ELEC). Puis sous ces répertoires on trouve des sous répertoires correspondant aux disques des machines du sous-réseau, le plus souvent le disque porte le même nom que la machine.

Exemple : CAL : debussy, campra, rameau, puccini, ..., milhaud plus les différents disques du serveur `/usr/local /opt /usr/X11 ... /usr/share`. En général les disques machines contiennent les fichiers des utilisateurs alors que les disques du serveur contiennent les logiciels et données utilisables par tous.

1.2 COMMANDES DE BASE

La suite de ce chapitre donne la liste des commandes de base, avec leurs principales options, nécessaires pour se déplacer dans l'arborescence, ainsi que les commandes de manipulation des fichiers et répertoires. Pour une description complète de ces commandes, utiliser la commande **man**

* * * **MAN** * * *

Commande	Options	Paramètres
man		Nom de la commande

Description : Retourne le mode d'emploi de la commande donnée en paramètre.

Paramètres : Nom de la commande.

Exemple : man man

1.2.1 Chemin d'accès à un élément de l'arbre

L'accès à un fichier ou à un répertoire se fera en donnant son **chemin (PATH)**.

Chemin absolu : tout chemin en absolu doit commencer par "/" (Racine de l'arbre) suivi des noms de répertoires correspondant aux noeuds, séparés par le caractère "/".

Chemin relatif:

désignation du répertoire courant: "."

désignation du répertoire père: ".."

Le point de départ sera le répertoire courant (pwd), puis le chemin se composera d'une succession de "." et de noms des répertoires séparés par "/" suivant que l'on descend ou remonte dans l'arborescence.

Exemple :

```
Je suis sous le répertoire ' /etc ' (Fig 1.4),  
je veux aller visualiser le contenu de /usr/include
```

```
ABSOLU: cd /usr/include
```

```
RELATIF: cd ../usr/include
```

1.2.2 Parcours de l'arbre

* * * **PWD** * * *

Commande	Options	Paramètres
pwd		

Description : retourne la référence absolue du répertoire courant (celui où l'on se trouve lors de l'exécution de la commande).

* * * **CD** * * *

Commande	Options	Paramètres
cd		Chemin d'accès au répertoire

Description : permet de se positionner sur le répertoire désigné.

Paramètres : le chemin du répertoire est désigné en absolu ou en relatif.

Exemple :

```
cd /usr/openwin/bin
cd ../../????
```

Remarques : si l'on omet le chemin d'accès au répertoire, on est positionné sous son **HOME directory** (répertoire sous lequel on est positionné lors du login).

1.2.3 Les répertoires

Les commandes relatives aux répertoires :

* * * **MKDIR** * * *

Commande	Options	Paramètres
mkdir		Nom du répertoire

Description : création d'un répertoire.

Paramètres : chemin d'accès au répertoire

* * * **RMDIR** * * *

Commande	Options	Paramètres
rmdir		Nom du répertoire

Description : supprime le répertoire si celui ci est vide.

Paramètres : chemin d'accès au répertoire à détruire.

* * * **LS** * * *

Commande	Options	Paramètres
ls	-aFiRlgt	Nom du répertoire

Description : liste le contenu d'un répertoire.

Options :

- **a** : ajoute à la liste, les fichiers cachés. Ce sont ceux dont le nom est précédé de ".". Il s'agit des fichiers d'initialisation (.profile, .bashrc, .login, .cshrc, .logout, .xinitrc ...).
- **F** : renseigne sur le type de fichier
 - * fichier exécutable
 - / c'est un répertoire
 - @ lien symbolique

- **i** : le fichier est précédé de son numéro d'inode. C'est la référence du fichier au niveau système.
- **R** : liste récursivement le contenu du répertoire désigné.
- **l** : liste les fichiers avec les renseignements suivants :
 1. Indique pour le propriétaire, le groupe et les autres, les droits en lecture "**r**", écriture "**w**", exécution "**x**". La première lettre indique s'il s'agit d'un répertoire "**d**" ou d'un lien symbolique "**l**" (voir commande "*ln*").
 2. nombre de liens sur le fichier.
 3. nom du propriétaire
 4. taille du fichier.
 5. date de création ou de modification
 6. nom du fichier.
- **g** : utilisée avec l'option "**l**", ajoute en colonne 4 le nom du groupe.
- **d** : renseigne sur les caractéristiques de "." au lieu de son contenu, à utiliser avec l'option **l**.
- **t** : liste les fichiers par date de modification au lieu de l'ordre alphabétique.
- **u** : liste les fichiers par date d'accès, à utiliser avec l'option **l**.
- **r** : Inverse l'ordre d'affichage des fichiers.

Paramètres : chemin d'accès au répertoire. Sans désignation de répertoire, c'est le répertoire courant "." qui est listé.

Exemples :

```

Se positionner sous la racine ( / ).
Lister son contenu avec l'option F.
Lister son contenu avec l'option "l".
Lister son contenu par ordre de date.
Idem ci-dessus mais en ordre inverse, le plus recent en fin de liste.

```

1.2.4 Les fichiers

Les commandes relatives aux fichiers :

* * * **CAT** * * *

Commande	Options	Paramètres
cat		Nom de fichier

Description : imprime le contenu du fichier sur la sortie standard.

En utilisant les redirections, on a les possibilités suivantes :

- `cat >fic1` : création du fichier fic1 et la commande attend les données du clavier, la fin de fichier sera <CTRL-D> sur une ligne vide. Attention les corrections ne sont valables que sur la ligne, après le **CR** plus de modifications possibles.

- `cat fic1 >> fic2`: ajout du contenu de `fic1` à `fic2`.

Remarques : si le fichier dépasse la taille de la page, on ne verra que la fin du fichier (voir commande suivante).

* * * **MORE** * * *

Commande	Options	Paramètres
more		Nom de fichier

Description : liste le contenu du fichier page par page.

Paramètres : nom de fichier.

Remarques : cette commande est souvent utilisée avec un **pipe** dans le style `:ls -l | more`

* * * **CP** * * *

Commande	Options	Paramètres
cp	-ipr	Source Destination

Description : copie un fichier (ou répertoire) source vers un fichier (ou répertoire) destination.

Options :

- **i** demande confirmation. Evite l'écrasement d'un fichier existant.
- **p** recopie en conservant la date et les droits d'accès.
- **r** recopie récursive. Si la source est un répertoire on recopie ce répertoire avec ses fichiers et ses sous répertoires.

Paramètres :

- `cp fic1 fic2` copie le fichier `fic1` dans `fic2`.
- `cp -r rep1 rep2` copie du répertoire `rep1` dans répertoire `rep2`.
Si `rep2` n'existe pas, il sera créé.
S'il existe `rep1` est copié dans `rep2`, `rep1` devenant un sous-répertoire de `rep2`.
- `cp fic1 rep1` recopie du fichier `fic1` dans le répertoire `rep1`, s'il existe uniquement.

* * * **MV** * * *

Commande	Options	Paramètres
mv		source - destination

Description : déplace les fichiers et répertoires dans le système de fichier. Cette commande peut être utilisée pour renommer un fichier ou un répertoire.

Options :

- **i**: mode interactif: évite l'écrasement d'un fichier existant.

Paramètres :

- `mv fic1 fic2` : renomme le fichier `fic1` en `fic2`. Si `fic2` existe, il est détruit.
- `mv rep1 rep2` : `rep1` est renommé en `rep2` si `rep2` n'existe pas, sinon voir forme suivante.
- `mv fic1 fic2 rep1 ...rep2` Déplace `fic1`, `fic2`, `rep1` et suivant dans `rep2`.

* * * **LN** * * *

Commande	Options	Paramètres
<code>ln</code>	<code>-s</code>	Source -Destination

Description : permet de créer un nouvelle référence pour un fichier existant. Physiquement il n'existe qu'un fichier, en fait on crée seulement une nouvelle entrée dans le répertoire (même numéro de inode).

Options : **s** : création d'un lien symbolique. On crée un lien vers un fichier existant dans un autre système de fichier, ou un lien entre répertoires, ce qui est interdit avec le lien sans l'option "s". Bien qu'il n'existe réellement qu'un fichier, on crée un nouvel inode.

Paramètres : nom de fichier existant - nom du fichier lié au précédent.

Remarques : à chaque création de lien sur un fichier, un compteur est incrémenté (`ls -l`). La destruction physique d'un fichier n'est effective que lorsque ce compteur de liens s'annule.

* * * **RM** * * *

Commande	Options	Paramètres
<code>rm</code>	<code>-fir</code>	Nom du fichier ou répertoire

Description : supprime la référence du fichier dans le répertoire. Si le compteur de liens arrive à zéro, alors il est physiquement détruit.

Options :

- **f** : force la commande sans considération de permission en écriture.
- **i** : interactif, demande confirmation avant l'exécution de la commande.
- **r** : destruction récursive à partir de la référence donnée.

Paramètres : nom du/des fichier(s), ou nom du répertoire.

Remarques : **ATTENTION !!!** cette commande peut être extrêmement dangereuse avec l'option `r` ou l'utilisation de métacaractères tel que `*`. Il n'y a pas de récupération possible s'il n'existe pas de sauvegarde.

* * * **CHMOD** * * *

Commande	Options	Paramètres
<code>chmod</code>	<code>-R</code>	Mode - Nom de fichier ou répertoire

Description : change les permissions en lecture, écriture, exécution.

Options :

- **R** : change les droits récursivement à partir du noeud donné en référence.

Paramètres : *modes* - référence des fichiers à modifier. Le mode peut avoir deux formes : absolu ou symbolique.

1. **Absolu** : en se référant à l’affichage des droits obtenus avec la commande `ls -l`, on a un champ de neuf bits découpé en trois champs de trois bits suivant le tableau ci-dessous.

U : utilisateur
G : groupe
O : autres

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

Chaque champ correspond dans l’ordre à :

le bit 1 **R** : de lecture
le bit 2 **W** : d’écriture
le bit 3 **X** : d’exécution

Le mode sera un nombre de 3 chiffres en octal, un bit à 1 correspond la validation de la permission correspondante, un bit à 0 supprime la permission.

Exemple : 731 (111 011 001) rwx pour l’utilisateur, rx pour le groupe et x pour les autres.

Un moyen pratique est, en se référant au tableau suivant, d’ajouter les champs correspondants aux droits désirés.

R	W	X
4	2	1

2. **Symbolique** : le mode a la forme suivante : *Qui +/-permission*

QUI : **u** pour l’utilisateur, **g** pour le groupe, **o** pour les autres et **a** pour tout le monde.

+/-permission : **r** pour read, **w** pour write et **x** pour execute.

***** FILE *****

Commande	Options	Paramètres
file		Nom du fichier

Description : retourne le type de fichier.

- ascii
- postscript
- exécutable sparce
- exécutable 68020

Exercices :

Dans le répertoire ‘‘~domst/TP-Shell/’’ vous trouverez les fichiers suivants:
Voir l’explication du tilde (~) dans le Chap 3.3.1 Redirections.
fic1 fic2 ... fica ...
Recopiez ce répertoire dans votre répertoire par la commande
‘‘cp -rp ~domst/TP-Shell .’’.

En utilisant les commandes précédemment décrites, réaliser les opérations suivantes~:

- Renommer le répertoire ‘‘TP-shell’’ en ‘‘rep’’
- Créer un nouveau répertoire rep1
- Positionner vous sous rep
- Recopier le contenu de rep dans rep1 en utilisant le meta caractere *.
 - cp * ../rep1
- Vérifier le contenu de rep et rep1, sans oublier les fichiers cachés, ces derniers n’ont pas ete recopies car l’* ne couvre pas le point (.) de debut de fichier. Detruire le contenu de rep1.
- Recommencer en utilisant la copie recursive et en creeant un nouveau repertoire rep2.
 - cp -r rep rep2
 - ls -al rep2
 - cd rep2
 - rm *
 - cd ..
 - rmdir rep2 (Cette commande n’est possible que si le repertoire est vide).
Si l’on veut détruire un répertoire non vide, on peut utiliser la récursivité.
 - rm -r rep2

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!! ATTENTION DANGER !!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!

- Positionner vous dans rep.
- Faire une copie de ‘‘fic1’’ et ‘‘fic2’’ dans ‘‘rep1’’
- Créer des liens de ‘‘fic3’’ et ‘‘fic4’’ vers ‘‘rep1/fic3’’ et ‘‘rep1/fic4’’
- Idem en changeant bien sûr de nom de destination par ex: ‘‘rep1/fic31’’.
- Vérifier les numéros d’inode dans les deux répertoires, ainsi que l’incrémentation du compteur de liens.
- Essayer de créer un lien, non pas sur un fichier comme ci-dessus, mais sur un répertoire
- ln rep rep-link (voir la commande ‘‘ln’’)
- Comparer les numéros d’inode

Chapitre 2

LES PROCESSUS

2.1 GENERALITES

2.1.1 Définition

Un fichier exécutable (programme compilé, script-shell) n'est qu'une suite d'octets stockée sur disque totalement inerte. Il ne deviendra processus que lorsqu'il est chargé en mémoire et pris en charge par le système d'exploitation.

2.1.2 Création

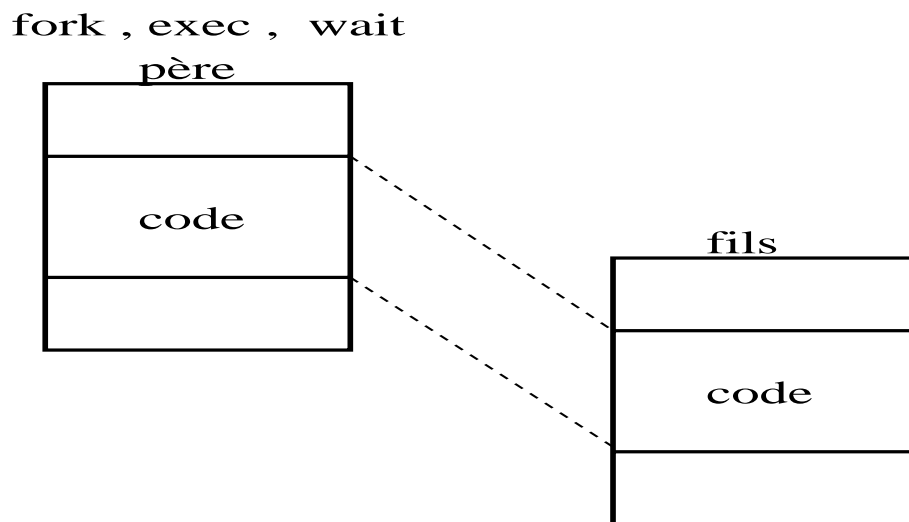


FIG. 2.1 – *Création*

Un des principes fondamentaux du système UNIX est la création de processus par le mécanisme de fourche (**fork**). Le principe en est le suivant:

Création d'un nouveau processus fils par duplication de la zone code d'un processus. Ce nouveau processus est le fils du processus père ayant appelé la fonction **fork**.

Remplacement de la zone code initiale dans le fils par le code d'un nouveau programme à l'aide de l'appel **exec**.

En principe le père, par l'appel à la fonction **wait** se met en attente de la fin du fils. Dans le cas où un père se termine avant son fils, ce dernier est rattaché au processus **init** de numéro 1, ancêtre de tous les processus du système.

2.1.3 Les entrées sorties

Sous UNIX les périphériques sont associés à des fichiers spéciaux référencés sous **/dev** (voir chapitre arborescence locale).

- **Clavier** : /dev/kbd
- **Console** : /dev/console
- **Fenêtre Xterm** : /dev/tty0
- **Souris** : /dev/mouse
- **Lecteur de disquettes**: /dev/fd0

D'autre part tout fichier ouvert sera référencé au niveau du processus par un numéro de **descripteur**. Ce numéro sera un petit entier 0,1,2,3,4 ... n .

A la création de tout processus, il y a ouverture automatique de trois fichiers spéciaux utilisés pour les entrées sorties.

- **stdin**: L'entrée standard par défaut le clavier de numéro de descripteur 0.
- **stdout**: La sortie standard par défaut la console de numéro de descripteur 1.
- **stderr**: La sortie erreur standard par défaut la console de numéro de descripteur 2.

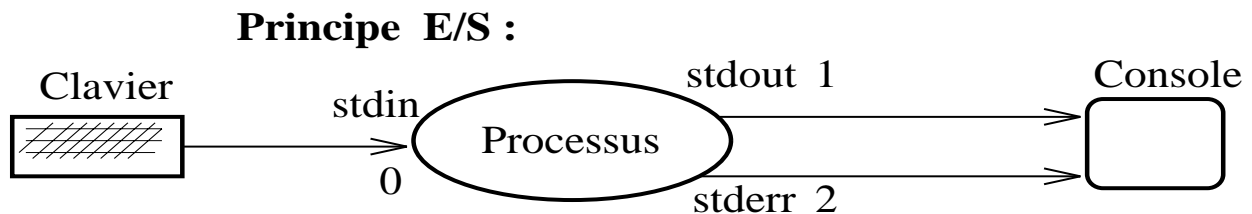


FIG. 2.2 – Entrées Sorties

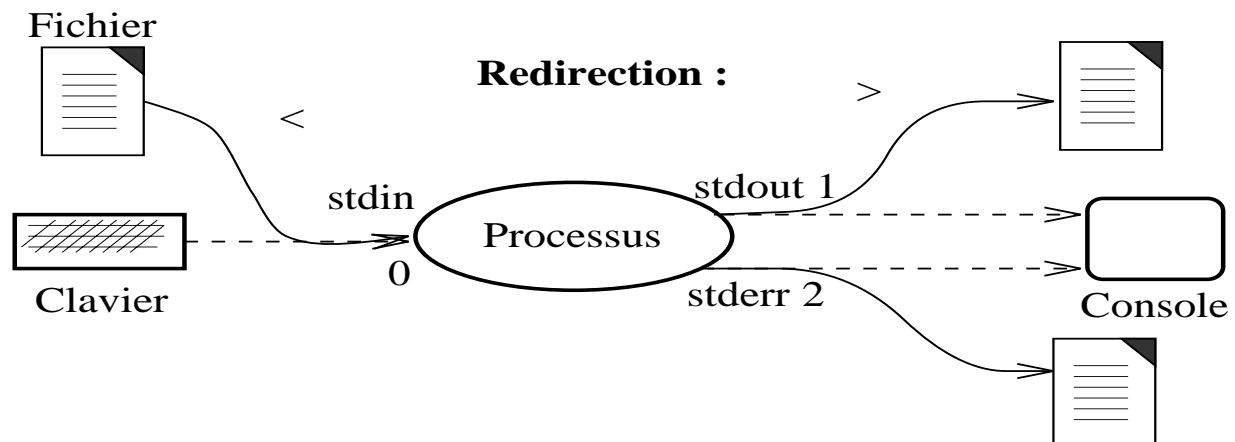


FIG. 2.3 – Redirections

Ce principe des entrées-sorties n'est pas figé. On a la possibilité de rediriger ces Entrées-Sorties. UNIX peut, par exemple fermer le fichier console de numéro de descripteur 1 et affecter

ce dernier à l'ouverture d'un nouveau fichier ou terminal, ainsi la sortie standard (stdout) est redirigée vers ce fichier suivant le schéma Fig 2.3. Cette opération se fera simplement au niveau de la ligne de commande comme nous le verrons dans le chapitre sur l'interpréteur de commande.

2.1.4 Communication

Par le mécanisme du fork, l'ensemble des processus du système UNIX forme une arborescence, et ils ne peuvent communiquer que par le biais de fichiers. Un moyen élémentaire de communication a été élaboré dès 1972 sur la seconde version d'UNIX, il s'agit de la communication par tube (**pipe**). C'est un mécanisme de file du type, premier entré premier sorti (**FIFO**). Bien que d'autres modes de communications aient été développés depuis, ce mécanisme est toujours fréquemment utilisé, notamment pour l'enchaînement des commandes élémentaires sous shell.

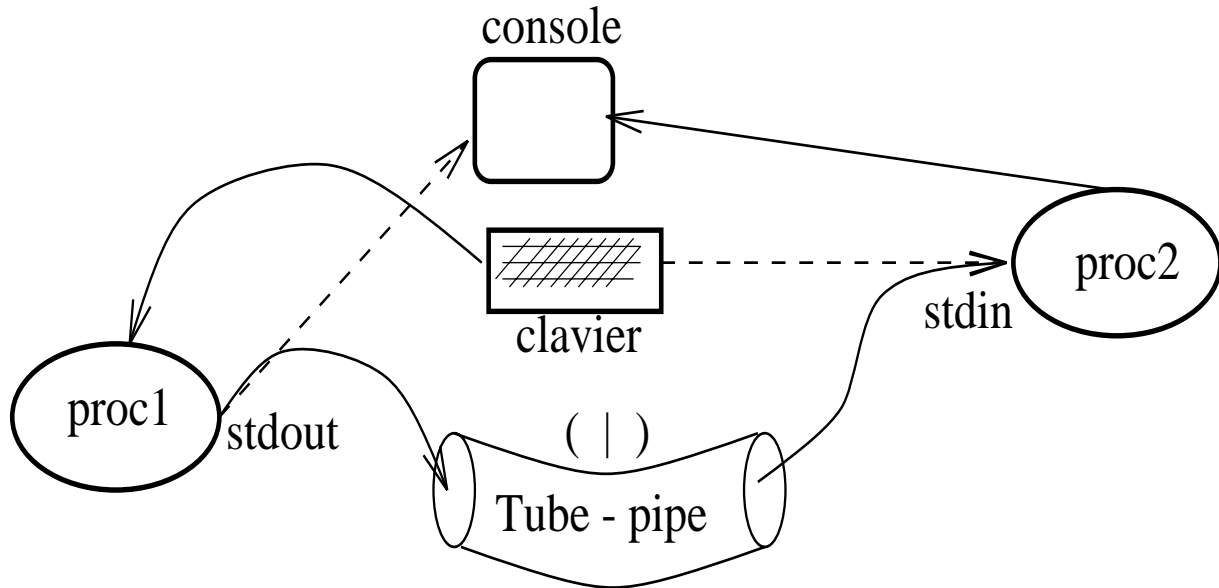


FIG. 2.4 – Tube

Un tube n'ayant pas de référence au niveau du système de fichiers, il n'est identifié qu'au niveau du processus par ses numéros de descripteurs. Ce qui entraîne que la communication ne peut avoir lieu qu'entre le processus créateur et ses descendants. C'est pour remédier à cette contrainte qu'on a créé les **tubes nommés**, comme leur nom l'indique, ils ont une référence de nom au niveau du système de fichiers, et peuvent être utilisés par tout processus pourvu que le propriétaire du processus en ait le droit (Fig 2.5).

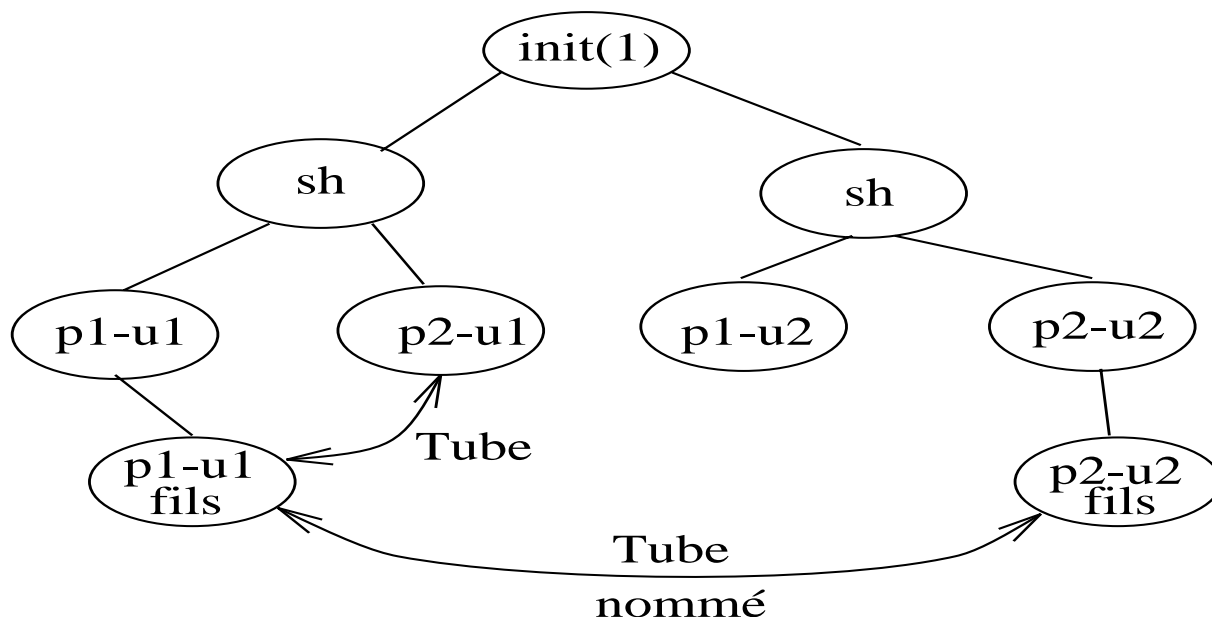


FIG. 2.5 – *Tube Nommés*

Pour améliorer la communication entre processus, d'autres moyens de communication ont été développés tant dans les souches **System V** que **BSD**.

IPC System V: Inter Process Communication

- **File de messages:** Un processus peut extraire ou déposer des messages dans une file suivant le principe d'une boîte aux lettres. Un message est un objet formant un tout manipulé en une seule opération, contrairement aux tubes qui permettent de gérer un flux caractère par caractère.
- **Mémoire partagée:** Partage par plusieurs processus d'une zone de mémoire commune.
- **Sémaphores:** Outil permettant à des processus de gérer l'accès concurrentiel à des ressources communes comme les files de messages et la mémoire partagée.

Socket BSD: Les outils décrits ci-dessus ne sont utilisables que par des processus sur un même système. Le système BSD possède le mécanisme de **Socket**, outil permettant la communication entre processus sur différentes machines.

2.1.5 Contrôle

2.1.6 Les signaux

Un utilisateur peut interrompre ses propres processus au niveau du shell en utilisant les **signaux** ci-dessous, soit à partir du clavier (*CTRL C*, *CTRL D*, *CTRL *), soit par la commande **kill**.

Signal	C	Clavier
1	SIGHUP	
2	SIGINT	ctrl-c
3	SIGQUIT	ctrl-\
9	SIGKILL	

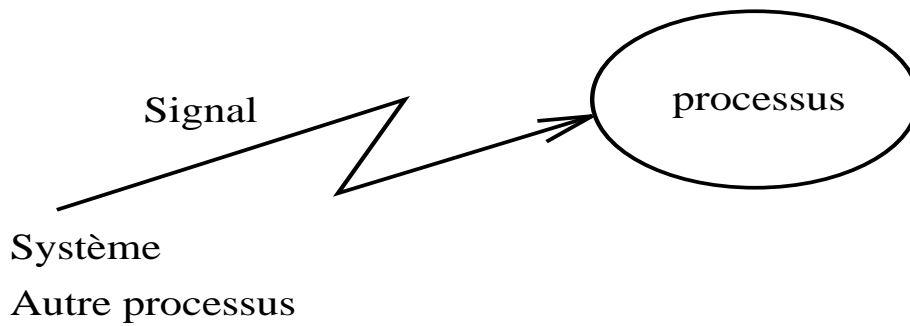


FIG. 2.6 – *Contrôle*

* * * **KILL** * * *

Commande	Options	Paramètres
kill	-l/n	numéro du processus

Description :

Options :

- **-l** s'utilise sans paramètre, retourne la liste des signaux disponibles sur la machine.
- **-n n** est le numéro du signal à envoyer.

Paramètres : numéro du processus récupéré par la commande **ps**.

Exercice:

En utilisant la commande `'sleep nn'` (nn étant un nombre de secondes), tester les signaux 2 (CTRL-C) et 3 (CTRL-`\`).

2.2 CARACTERISTIQUES

2.2.1 Type

On distinguera deux principaux types de procesus qui sont les suivants:

Système (daemon) Appartenant au super utilisateur **root**, ils possèdent les caractéristiques suivantes:

- Pas de terminal d'attachement.
- Exécuté au lancement du système d'exploitation (**boot**).
- Exécuté à une date précise (Super-user cron).

Utilisateurs Procesus propres à chaque utilisateur: Commandes UNIX, Programmes compilés

2.2.2 Identité

Les principaux paramètres permettant d'identifier un processus sont les suivants:

- Identification (PID - nom)
- Propriétaire (UID - GID)
- Terminal d'attachement.
- Fichiers ouverts (stdin:0, stdout:1, stderr:2)

2.2.3 Etat

- Runnable R
- Stoppé T
- Swappé sur disque W
- Zombie Z
- Endormi S
- Non interruptible D

2.2.4 Mode d'exécution

2.3 LA COMMANDE ps

Systeme BSD :

Exercice :

```
Avec les commandes relatives aux processus décrites ci-dessous,
essayez et interprétez les commandes suivantes~:
```

```
- ps
- ps -l
- sleep 60 &
- ps -lx
- rlogin "machine"
- ps
- ps -l
- ps -al
- ps -au
- ps -ax | more
- ps -axu | more
- ps -axl | more
- ps -tco
```

```
où co est la console et p0 le pseudo-terminal ttyp0 ...
```

* * * **PS** * * *

Commande	Options	Paramètres
ps	-axul	

Description : Liste des processus. Sans option, on liste ses propres processus.

Options :

– **l** : Description complète du processus.

F : drapeaux informant si le processus :

- est en mémoire
- swappé sur disque
- effectue des E/S
- ...

UID : numéro d'identification de l'utilisateur.

PID : numéro de processus.

PPID : numéro de processus père.

CP PRI ... WCHAN : différents renseignements sur le temps CPU, priorité, taille mémoire occupée ...

STAT : état du processus

- **R** en cours d'exécution.
- **T** processus stoppé.
- **I** processus endormi (>20s).
- **S** processus endormi (<20s).
- **Z** processus zombie.
- **D** processus non interruptible.
- **W** processus swappé sur disque.

TT : terminal de rattachement du processus. Ceux qui n'en ont pas, tels les démons système, sont identifiés par “?”

TIME : temps d'exécution.

Command : nom du processus.

- **x** : liste le processus n'ayant pas de terminal associé. Ils sont repérés par “?” sous la colonne **TT**, en particulier les démons système et les processus rattachés à **init** (proc-num=1).
- **u** : spécifie le nom du propriétaire du processus.
- **a** : liste les processus détenus par les autres utilisateurs.
- **t** : liste uniquement les processus attachés à un terminal particulier (ex: **ps -axtco**).

Systeme V :

Exercice:

Avec les commandes relatives aux processus decrites ci-dessous, essayez et interpretez les commandes suivantes :

```
- ps
- ps -f
- ps -l
- ps -e | more
- ps -ef | more
- ps -el | more
- rlogin "machine"
- ps
- ps -f
- ps -t "terminal"
- ps -u "nom ou UID du proprietaire"
```

***** PS *****

Commande	Options	Paramètres
ps	-aelf	

Description : Liste des processus. Sans option, on liste les processus associés au terminal ayant lancé la commande **ps**. Seules les informations suivantes seront affichées: **PID** Identification du processus, **TTY** Identification du terminal, **TIME** Temps d'exécution cumulé, **COMD** Le nom de la commande. Pour plus d'informations, utiliser les options ci-dessous. Certaines options demandent une listes d'arguments. Cette liste sera une suite d'arguments séparés par des virgules (arg1,arg2,arg3), ou une suite d'arguments séparés par des blancs mais entre doubles quotes (" arg1 arg2 arg3 ")

Options :

– **f** ou **l**:Description plus ou moins complète du processus.Ces deux options (f ou l) signifiant respectivement Full ou Long, modifient le type d'informations retournées pour chaque processus, et non le type de processus à lister. Dans la liste des paramètres ci-dessous sera précisé entre () à quel type d'options correspond le paramètre affiché, f ou/et l.

F : (l) Drapeaux associés aux processus conservés pour des raisons historiques, ne leur attribuer aucune signification particulière :

UID :Numéro d'identification de l'utilisateur.

PID :Numéro de processus.

PPID :Numéro de processus père.

CP PRI ... WCHAN : Différents renseignements sur le temps CPU, priorité, taille mémoire occupée ...

STAT :Etat du processus

– **O** En cours d'exécution.

- **R** En attente d'exécution.
- **T** Process stoppé (¡CTRL Z¡).
- **I** process endormi .
- **S** Process endormi en attente d'événement.
- **Z** Process zombi (processus terminé n'ayant plus de père).

TT : Terminal de rattachement du processus. Ceux qui n'en ont pas, tels les démons systèmes, sont identifiés par "?"

TIME : Temps d'exécution.

STIME : Heure d' exécution.

Command : Nom du processus.

- **e** : Liste tous les processus
- **p** : Liste le ou les processus dont le numéro suit l'option p.
- **t** : Liste les processus attachés au terminal donné en paramètre.
- **u** : Liste uniquement les processus appartenant au propriétaire dont le nom ou l'UID est donné en paramètre.

Chapitre 3

L'INTERPRETEUR DE COMMANDES

3.1 GENERALITES

L'interpréteur de commande, premier processus d'un utilisateur lors de l'ouverture d'une session, est l'interface de communication entre ce dernier et le système d'exploitation. On l'utilisera suivant les deux modes suivants:

Mode commande: Dans ce mode interactif et conversationnel, l'utilisateur demande l'exécution de ses commandes au coup par coup, avec possibilité d'enchaîner quelques commandes sur la même ligne (**pipe ...**).

Mode programmation : Dans ce mode le **shell** devient un véritable langage de programmation. Les programmes sont constitués d'une suite de commandes simples, de définition de variables, contrôlées par des instructions (**test**, **itération ...**).

Le **Shell** est paramétrable grâce à ses fichiers d'initialisation (**.login**, **.profile**, **.bashrc**, **.zshrc ...**) permettant de définir l'environnement d'un utilisateur, grâce à des variables standard prédéfinies (**PATH**, **HOME**, **TERM ...**).

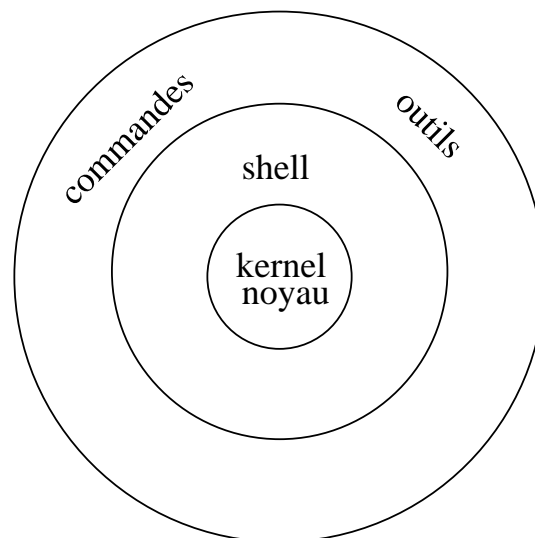


FIG. 3.1 – *Interpréteur de commandes*

3.2 OUVERTURE D'UNE SESSION

login :

passwd :

Prompt: < *gadret @ sirocco: 123* >

```
gadret:9Ca3/sjsBBh.:10001:110:Serge Gadret,C218:/inf/sirocco/admin/gadret:/bin/bash
```

Nom de login	UID	Identite	Le programme exécuté à la connexion
	GID	Chemin d'accès au répertoire de l'utilisateur	SHELL
Password Encodé			

FIG. 3.2 – Ouverture d'une session

A l'ouverture d'une session, la procédure de **login** va vérifier votre identité ainsi que votre mot de passe dans le fichier **passwd**. Dans ce fichier chaque ligne, représentée sur la Fig 3.2 regroupe les différentes informations sur un utilisateur, nécessaires à la gestion de celui-ci par le système. Ces informations sont les suivantes:

login Nom de l'utilisateur utilisé pour l'ouverture d'une session.

passwd Mot de passe chiffré.

UID Valeur numérique correspondant au nom de login, c'est cette valeur qui est utilisée par le système pour identifier le propriétaire des fichiers (inode) et processus.

GID Idem ci-dessus au niveau du groupe d'appartenance d'un utilisateur.

Commentaire Zone commentaires, en principe nom, prénom, tél ... de l'utilisateur.

PATH Chemin d'accès au répertoire de l'utilisateur, correspond à la variable **PATH** du shell.

Shell Nom du shell qui sera lancé à la connexion.

3.3 LA LIGNE DE COMMANDE

La ligne de commande représentée Fig 3.3 se compose d'un ou plusieurs champs terminés par un retour chariot (**return**).Le séparateur de champs est par défaut le blanc ou tab.

Après la procédure de connexion, vous êtes sous contrôle d'un interpréteur de commandes **/bin/csh** ou **/usr/local/bin/enstsh** qui vous invite à entrer vos commandes après un "*prompt*".

```
<gadret@valjean: 34>
```

Le "*prompt*" se compose du nom de login *gadret* suivi du nom de machine *valjean* suivi du numéro de commande *34*. Ce "*prompt*" est paramétrable comme nous le verrons dans le chapitre

sur le **Shell**.

Les généralités citées ci-dessous (excepté l'utilisation du “~” sous **/bin/sh**) s’appliquent à tout autre interpréteur de commande (Bourne-Shell, Korn-Shell, Bash-Shell ...).

COMMANDE	-OPTIONS	PARAMETRES	REDIRECTIONS
Mnémonic (<i>ls, ps, mv ...</i>)	Suite de lettres	Nom de fichier répertoire programme machine adresse réseau	< Entrée standard (0) > Sortie standard (1) >> ajout 2> Sortie Erreur (2) 2>&1 Sortie erreur = sortie standard
Nom <i>mail</i>	<i>-algF</i>		fichier Autre terminal <i>/dev/null</i>
Exemples: <i>ps</i> <i>ls -lag</i> <i>cp fic1 fic2</i> <i>awk -F':' '/gadret/ {print \$1 \$6}' /etc/passwd</i> <i>telnet zeus.enst.fr</i> <i>mail -s sujet dax@inf.enst.fr < message</i>			

FIG. 3.3 – La ligne de commande

Commande : mnémonique de la commande : (*ls, mv, cp, rm, ...*)

Les commandes utilisées ci-dessous sont décrites dans les chapitres précédents:

- **cd** Change Directory, change de répertoire courant.
- **ls** LiSt, liste le contenu d’un répertoire.
- **ps** Process Statut, liste les processus et leur état.
- **cp** CoPy, recopie de fichiers.

Options : elles débutent généralement par un tiret “-” et sont composées d’une suite de lettres, permettant de modifier son comportement de base.

Exemples :

```
(gadret@valjean: 190) cd /
(gadret@valjean: 190) ls
(gadret@valjean: 190) ls -lag
(gadret@valjean: 190) ps
(gadret@valjean: 190) ps -aux
```

Certaines commandes possèdent des options paramétrables comme dans l’exemple suivant:

```
(gadret@valjean: 190) find . -name toto -depth -ctime 2 -print
```

Paramètres : variables suivant les commandes. Dans la majorité des cas on trouve des noms de fichier ou de répertoire, mais également des noms de machine, des adresses réseau, des lignes d’instructions (**sed, awk**).

3.3.1 Redirections

Redirections : le shell reçoit vos commandes du clavier qui est l'*entrée standard* par défaut et renvoie les résultats de la commande sur la *sortie standard*, écran du terminal par défaut, juste après la connexion c'est en principe la console.

Il est cependant possible de rediriger les entrées / sorties :

- Redirection de l'entrée : < "autre terminal ou fichier"
- Redirection de la sortie : > "autre terminal ou fichier"
- Redirection de la sortie avec ajout dans un fichier existant : >> "fichier"
- Redirection de la sortie erreur : 2 > (sh, bash, zsh) "autre terminal ou fichier"

- Redirection de la sortie erreur idem à la sortie standard : 2 > &1

Remarque : Tilde (~)

On remarquera l'utilisation de "~" (tilde) qui remplace le chemin absolu d'accès au répertoire principal d'un utilisateur. Ici l'utilisateur **gadret** a pour "Home-directory" `/inf/ulyse/admin/gadret`, donc "~" = `/inf/ulyse/admin/gadret`. Si on veut référencer son propre home-directory, "~" suffit.

Exercice :

```
(gadret@valjean: 190) cd /
(gadret@valjean: 191) ls -lag >~/toto
(gadret@valjean: 192) cd ~
(gadret@valjean: 193) cat toto
(gadret@valjean: 194) ls -l >>toto
(gadret@valjean: 195) more toto
(gadret@valjean: 195) cd
Verifier que le fichier toto existe et que tyty n'existe pas.
executer les commandes suivantes et interpreter.
(gadret@valjean: 195) ls -l toto tyty
(gadret@valjean: 195) ls -l toto tyty >stdout 2>stderr
(gadret@valjean: 195) cat stdout
(gadret@valjean: 195) cat stderr
(gadret@valjean: 195) rm stdout stderr
(gadret@valjean: 195) ls -l toto tyty >stdout 2>&1
(gadret@valjean: 195) cat stdout
(gadret@valjean: 195) rm stdout
```

3.3.2 Enchaînement de commandes

Séquentiel Il est possible de regrouper plusieurs commandes sur une même ligne. Pour cela on sépare chaque commande par le caractère ";" (point virgule).

Exemple :

```
(gadret@valjean: 44) date > heure; sleep 10; cat heure
```

La commande *date* renvoie la date et l'heure que l'on sauvegarde dans le fichier *heure*, puis avec *sleep* on attend 10 secondes, enfin on liste le contenu du fichier avec la commande *cat*. En fait le **shell** exécute ces commandes les unes après les autres de façon indépendante.

Tubes (|) Une autre méthode consiste en l'utilisation de tube (**PIPE**).

Syntaxe: `cmde1 | cmde2 | cmde3`

La sortie de `cmd1` sera dérivée de la sortie standard (console) vers l'entrée de la commande suivante `cmd2`, sa sortie sera renvoyée à son tour sur l'entrée de `cmd3`.

Exercice :

```
ls -l /usr/bin | more
```

On liste le contenu de */usr/bin* qui dépasse la taille de l'écran, ainsi en renvoyant la sortie sur l'entrée de la commande *more* on obtient un listing page par page.

3.3.3 Mode d'exécution

Background

Lorsque l'on exécute une commande, notamment une compilation qui demande beaucoup de temps CPU, on lance cette commande en arrière plan. Le shell vous renvoie le prompt immédiatement, ainsi vous pouvez exécuter plusieurs commandes simultanément. Pour ce faire il suffit de terminer la ligne de commande par un `&`.

Exercices :

```
- sleep 120 &
- ls -lR /usr/lib | wc -l > nb-fic &
- ps -xl
- Toujours à l'aide de la commande 'sleep nn', mais exécutée
  en background, récupérer avec ps le numéro de processus,
  puis tester les signaux 2-3-9 avec la commande kill.
```

3.3.4 Méta-caractères

Caractères spéciaux : Les caractères spéciaux ont pour but de simplifier l'écriture des commandes, principalement les références de fichiers ou de répertoires. Les principaux méta-caractères sont :

- `*` : remplace n'importe quelle chaîne de caractères. Si vous voulez référencer des fichiers cachés (dont le nom commence par ".") le point n'est pas couvert par `*`.

- ? : remplace un caractère quelconque et un seul.
- [...]: désigne un caractère quelconque dans l'ensemble défini entre crochets. Quand il s'agit d'un ensemble tel que 012...89 ou abcd...xyz on utilise la notation suivante: [1-9] [a-z]
- \ : Permet d'enlever à tous ces caractères leur sens de meta-caractère en les précédant de backslash .

Exercice :

```

Dans le répertoire ‘‘~domst/TP-Shell/’’ vous trouverez les
fichiers suivants:
  fic1 fic2 ... fica ...
Recopiez ce répertoire par la commande ‘‘cp -rp ~domst/TP-Shell .’’ .
En utilisant les méta-caractères lister les caractéristiques
des fichiers ( ls -l ...) suivants~:
- Tous les fichiers.
- En une seule commande les fichiers commençant par ‘‘f’’.
- On peut vérifier que ‘‘*’’ ne couvre pas le ‘‘.’’ (ls -la).
- Idem mais en utilisant le ‘‘?’’, uniquement les fic...
- Enfin uniquement les fichiers terminés par un chiffre.
Utilisation du backslash :
- echo coucou > toto\?
- ls
- cp toto\? toto*
- cp toto\? toto\*
- ls
- rm toto?

```

3.4 BOURNE SHELL

Le **Bourne Shell** (/bin/sh) est le shell d'origine d'UNIX (1970), ce shell est très peu convivial, il est délaissé au profit de shells tels que **ksh**, **bash**, **zsh** pour l'utilisation en mode commande, ils sont entièrement compatibles avec le bourne shell. Par contre en mode programmation, bien que les nouveaux shells apportent de nouvelles facilités dans ce mode, le Bourne Shell reste bien suffisant dans la majorité des cas et de plus, étant présent sur tout système UNIX, rendra tout script portable quelque soit le système.

3.4.1 Les variables du sh

Définition :

- **nom=valeur** : affecte valeur à la variable d'identificateur nom. Attention nom et valeur sont accolés au signe =.
- **read var1 var2 var3** : provoque la lecture d'une ligne au clavier, chaque mot sera affecté à une variable.

- **unset nom** : supprime la variable.
- **env**: liste les variables qui seront transmises aux processus descendants. Ce sont les variables exportables.
- **export var** rend la variable locale “var” exportable. Sans argument donne la liste des variables locales devenues exportables.

Remarques: il est à noter que les variables d’environnement PS1 et PS2 ne sont pas exportées automatiquement.

Une modification locale d’une variable d’environnement ne sera pas répercutée dans les processus ascendants.

Variables standards :

PATH: définit les différents répertoires dans lesquels seront recherchés les commandes exécutées par l’utilisateur.

TERM: définit le type de terminal (sun-cmd, vt100 ...)

PS1: définit la valeur du premier prompt. Par défaut ce sera “\$”

PS2: définit la valeur d’un second prompt. Par défaut ce sera “>”

Substitution de variable :

Pour faire référence à une variable il faut faire précéder le nom de la variable par le symbole “\$”.

echo “\$nom”: affiche le contenu de la variable “nom”.

Exercice :

```
<gadret@valjean: 1> sh
$ toutou=medor
$ echo $toutou
medor
- Définir plusieurs variables en une ligne de commande.
- Vérifier leur affectation.
- Lancer un nouveau shell (/bin/sh).
- Visualiser vos variables.
- Les rendre exportables en revenant au shell precedent par exit.
- Recreer un nouveau shell, pour verifier quelles sont bien
exportees.
- modifier une ou plusieurs variables.
- Verifier qu’elles ne sont pas modifiees dans le shell d’origine.
```

3.4.2 Fichiers d’initialisation

.profile: ce fichier sert principalement à l’initialisation de l’environnement (PROMPT, PATH, TERM) au lancement du shell.

3.4.3 Traitement des signaux

La réception d’un signal par un processus entraîne par défaut la terminaison de celui-ci. Mais on a la possibilité soit de masquer, soit de le dérouter avant la fin du processus.

Sous shell, la commande **trap ‘cmde’ n1 n2 n3** permet l’exécution de la commande *cmde* à la réception des signaux *n1 n2 n3*.

Sans arguments la commande *trap* permet de visualiser les signaux dérouterés.

La commande *trap n* permet de réactiver le signal n.

Exemples:

- Bloquer le signal 2.
- Dérouter le signal 3 par exemple vers 'echo "signal 3 dérouté"'
- Vérifier l'état de vos signaux
- Tester ces signaux avec la commande "sleep nn".
- Restituer leur comportement de base.

3.5 LE ZSH

Le **zsh** compatible **sh** possède, en plus de ce dernier des fonctionnalités rendant son utilisation plus souple et performante. Les plus importantes étant l'historique avec rappel des commandes, en mode édition, mécanisme d'alias.

3.5.1 Les variables du zsh

Comme dans le Bourne Shell, il est possible de définir des variables de deux types. Les variables locales au zsh et celles qui sont exportables. Certaines sont standards au zsh et identiques au Bourne Shell (**PATH**, **PROMPT**, **TERM**).

Définition :

Identique au Bourne Shell du paragraphe précédent.

Variabes standard :

PATH : Idem au Bourne Shell

TERM : Idem au Bourne Shell

PROMPT : définit la valeur du prompt.

HISTSIZE : définit le nombre de commandes conservées dans l'historique.

HOSTNAME : Correspond au nom de la machine sur laquelle est exécuté le shell.

UID : Variable égale à l'uid de l'utilisateur défini dans le fichier passwd.

SHELL : Correspond au nom du shell courant.

MAILCHECK : Définit en secondes la fréquence de test de l'arrivée de nouveaux mails.

Visualisation :

Idem au Bourne Shell.

3.5.2 Fichiers d'initialisation

Lors du lancement de **zsh** les fichiers suivants seront exécutés:

~/.zshenv :

~/.profile : Il sera exécuté s'il s'agit d'un shell de login.

~/.zshrc : Est exécuté pour un shell interactif.

~/.zlogin : Est exécuté également pour un shell de login.

3.5.3 Alias

les commandes *alias* et *unalias* permettent respectivement de définir de nouvelles commandes ou de les supprimer. Ce mécanisme est principalement utilisé pour simplifier l'écriture de commande ou enchaînement de commandes par utilisation d'abréviation.

Syntaxe :

alias *nom d'alias* = '*commande*'

unalias '*nom d'alias*'

exemple :

```
(gadret@sirocco 14) alias lg='ls -lgF'  
(gadret@sirocco 14) alias bye=logout  
En général les alias sont définis dans le .zshrc.
```

3.5.4 Historique des commandes

Le principe de cet historique est de pouvoir mémoriser et de rappeler d'anciennes commandes. Le nombre de commandes mémorisées est défini par la variable *HISTSIZE*.

Les commandes sont les suivantes :

history : Liste les dernières commandes exécutées. Souvent redéfini par *alias* en **h**.

! **“num”** : Rappel de la commande de numéro “num”.

!! : Rappel de la dernière commande.

! **“chaîne”** : Rappel de la commande la plus récente commençant par “chaîne”.

De plus le **zsh** vous permet d'utiliser les touches ↑ et ↓ pour vous déplacer dans l'historique en mode édition, les touches ← et → vous permettent de vous déplacer sur la ligne de commande vous permettant ainsi de la modifier.

3.5.5 Complètement des noms de fichiers

En utilisant la touche **Tab** il est possible de compléter les noms de fichiers dont vous avez tapés les premières lettres.

3.6 L' ENVIRONNEMENT ENST

En vous référant à la Fig 3.4 suivante vous disposez des renseignements sur un environnement généré à la suite de la procédure **install-startup** avec les paramètres suivants:

- Système de multifenêtrage: X11
- Gestionnaire de fenêtre: mwm
- Shell de travail: zsh

Vous pourrez vérifier vos propres paramètres en visualisant le fichier généré par **install-startup** **.shenv.preload**. Ce fichier crée automatiquement ne doit pas être modifié manuellement.

On distingue deux espaces distincts principaux :

- Utilisateur: \$HOME
- Système: /usr/local/startup (\$STARTUP)

Le déroulement des opérations au cours d'une ouverture de session sera le suivant:

Le shell de login (/bin/sh) exécute le fichier d'initialisation **.profile** situé dans le repertoire de l'utilisateur, qui a son tour exécute le fichier système **\$STARTUP/profile**.

\$STARTUP/profile exécutera les étapes suivantes:

- Lecture de \$HOME/.shenv.preload pour déterminer les choix de l'utilisateur (Multifenêtrage, Window-manager, Shell)

- Exécution de \$STARTUP/shenv pour configurer les variables d'environnement (PATH ARCH SHELL EDITOR ...).
 - Exécution de \$HOME/.shenv s'il existe, pour compléter l'environnement propre à chaque utilisateur.
- Lancement du système de multifenêtrage (startx).

Le démarrage du système de multifenêtrage va exécuter le fichier d'initialisation \$HOME/.xinitrc qui générera les étapes suivantes:

- Lecture des fichiers de configuration de X11 (Xdefaults,Xmodmaprc ...)
- Exécution des outils X suivants: deux fenêtres xterm (xconsole sera non interactive). le gestionnaire de fenêtre (mwm).

A chaque Xterm sera attaché un shell, zsh qui ira à son tour lire son fichier d'initialisation \$HOME/.zshrc, ce dernier configurera les variables propres au zsh ainsi que les alias.

Exercice :

- Pour suivre le déroulement des opérations vous pouvez dans les fichiers vous appartenant (.profile .shenv .zsh), à l'aide de la commande echo envoyer des messages à la console.
exemple: echo 'debut .profile'
- Les fichiers grisés sur la figure sont ceux qui vous permettent de compléter votre environnement (.shenv,.xinitrc,.zshrc)
Définir une variable d'environnement, définir ou redéfinir des alias,
Ajouter de nouveaux outils X (xeyes, xclock ...) au lancement de X.

L ' ENVIRONNEMENT ENST

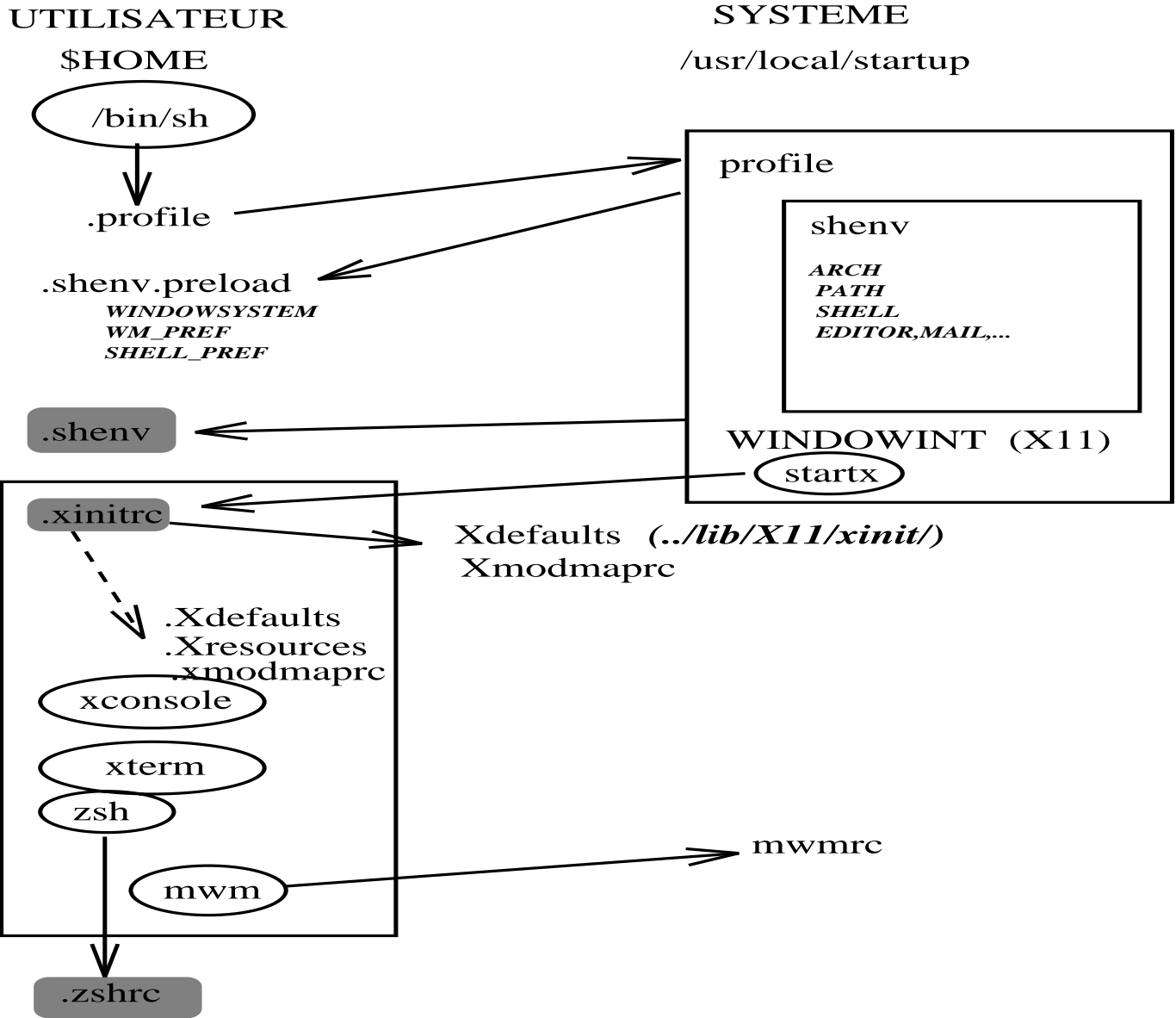


FIG. 3.4 – Environnement ENST

Chapitre 4

COMMANDES UNIX

4.1 COMMANDES ELEMENTAIRES

***** logname *****

Commande	Options	Paramètres
logname		

Description : Retourne le nom de login de l'utilisateur.

Exemples :

```
valjean $ logname
gadret
```

***** who *****

Commande	Options	Paramètres
who [am i]		

Description : Retourne le nom des utilisateurs qui ont ouvert une session. On a dans l'ordre les champs suivants: Le nom de login, le nom du terminal , l'heure de connection. La commande **who am i** liste uniquement l'utilisateur ayant tapé la commande.

Exemples :

```
zenon ~ $ who
farhat    console      sep  25 14:00
gadret    pts/8           sep  25 14:14 (valjean.enst.fr)
farhat    pts/4           sep  25 14:00
farhat    pts/5           sep  25 14:00
farhat    pts/6           sep  25 14:00

zenon ~ $ who am i
gadret    pts/8           sep  25 14:14 (valjean.enst.fr)
```

***** id *****

Commande	Options	Paramètres
id		

Description : Retourne le nom de login , l'UID, le groupe et le GID .

Exemples :

```
chimene ~ $ id
uid=10001(gadret) gid=110(inf)
```

***** W *****

Commande	Options	Paramètres
w	-hs	nom d'utilisateur

Description : Cette commande donne un aperçu de l'activité de la machine.

L'entête liste respectivement l'heure, depuis combien de temps le système est actif, le nombre d'utilisateurs connectés et la charge du système représentée en nombre de taches en attente depuis 1, 5, 15 minutes.

Les renseignements sur les utilisateurs connectés sont les suivants:

nom de login, terminal de connexion, l'heure de connexion, le temps depuis lequel l'utilisateur n'a pas utilisé son terminal en heures et minutes, le temps CPU utilisé par tous les processus et leurs descendants, le temps CPU du processus courant actif, le nom du processus courant.

Options :

- **h** : Supprime l'entête.
- **s** : Affichage restreint aux champs suivants:
Nom d'utilisateur, heure de connexion, temps depuis que lequel l'utilisateur n'a pas utilisé son terminal en heures et minutes, le nom du processus courant.

Paramètres nom d'utilisateur: Affiche uniquement l'utilisateur désigné.

Exemples :

```
ulyссе ~/new_poly/new $ w
 3:11pm up 19 days, 6:28, 5 users, load average: 1.79, 3.29, 3.72
User   tty      login@  idle   JCPU   PCPU   what
gadret ttyp0    8:32am  10     3      1     /usr/local/bin/zsh
gadret ttyp2    10:58am 10     46     5     /usr/local/bin/zsh
rungsawa ttyp5    1:56pm  10     18     5     -tcsh
dax     ttyp6    Thu 3pm  27     2:56   2:54   xload -geometry 75x75+470+05 -fg
dax     ttyp7    Thu 3pm  2       2     1     /usr/local/bin/zsh
```

```
ulyссе ~/new_poly/new $ w -s
 3:11pm up 19 days, 6:28, 5 users, load average: 3.42, 3.61, 3.84
User   tty  idle  what
gadret p0   10   zsh
gadret p2           zsh
rungsawa p5   10   tcsh
dax     p6   27   xload
dax     p7           zsh
```

*** rlogin ***

Commande	Options	Paramètres
rlogin	-l nom de login	nom de machine

Description : Permet d'ouvrir une session à partir de votre terminal vers une machine distante.

Options :

– **-l nom de login :** Permet d'ouvrir la session distante sous une identité différente.

Paramètres nom de machine : Désigne la machine sur laquelle on souhaite ouvrir une session.

Remarques : On peut couper la connexion avec la séquence suivante: `~.`, attention il ne s'agit pas d'un logout normal mais d'une rupture de connexion.

Exemples :

```
valjean ~ $ who am i
gadret pts/1 Sep 25 08:24
valjean ~ $ rlogin -l busch noroit
Password:

noroit ~ $ pwd
/inf/khamsin2/infas/busch
```

***** telnet *****

Commande	Options	Paramètres
telnet		nom de machine

Description : Permet d'établir une connexion avec une machine distante. Il n'y a pas d'ouverture automatique d'une session, vous devez, comme pour une session en local déclarer votre identité ainsi que votre passwd.

Paramètres nom de machine : Nom de la machine sur laquelle on souhaite ouvrir une session.

Exemples :

```
valjean ~ $ telnet khamsin
Trying 137.194.160.10...
Connected to khamsin.enst.fr.
Escape character is '^]'.
```

```
SunOS UNIX (khamsin)
```

```
login: gadret
Password:
khamsin ~ $
```

***** date *****

Commande	Options	Paramètres
date		

Description : Retourne la date .

Options : Le format d'affichage est modifiable, voir le manuel (man date).

Exemples :

```
ulyse ~ $ date

Mon Sep 25 16:12:37 MET 1995
```

***** cal *****

Commande	Options	Paramètres
cal		mois année

Description : Affiche le calendrier de l'année spécifiée en paramètre. Si on précise un mois (chiffre entre 1 et 12) affiche uniquement le mois. Si on ne donne aucun paramètre, affiche le mois courant de l'année courante.

Paramètres : mois (1-12), année sur 4 chiffres

Exemples :

```
[root@valjean: 110] cal 11 95
November 95
S M Tu W Th F S
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
## Il s'agit le 1'année 0095 et non 1995 ##
```

*** * * sleep * * ***

Commande	Options	Paramètres
sleep		temps en secondes

Description : Processus qui ne fait rien pendant n secondes. Cette commande est utilisée comme temporisateur, par exemple pour exécuter une commande après un certain délai.

Paramètres : Temps en secondes.

Exemples : Envoi d'un bip sonore chaque seconde.

```
while true
do
echo "^G"
sleep 1
done
```

*** * * tty * * ***

Commande	Options	Paramètres
tty		

Description : Permet de connaître le nom du terminal .

*** * * echo * * ***

Commande	Options	Paramètres
echo	-n	suite de paramètres

Description : Envoie en écho sur votre terminal les paramètres spécifiés à la suite de la commande.

Options :

- **n** : Supprime le retour chariot en fin de commande.

Paramètres : Suite d'arguments qui seront affichés sur le terminal séparés par un blanc. Si l'on veut afficher du texte mettre la chaîne entre simples quotes.

Exemples :

```
echo 'Module station de travail'
      'Module station de travail'

echo Module station de travail
      Module station de travail
```

*** **uname** ***

Commande	Options	Paramètres
uname	-anrs	

Description : Donne sans option le nom du système d'exploitation.

Options :

- **a** : Affichage des informations complètes sur la machine.
- **n** : Affiche le nom de la machine.
- **r** : Affiche la release du système d'exploitation.
- **s** : Affiche le nom du système d'exploitation.

Exemples :

```
typhon ~ $ uname -a
SunOS typhon 4.1.3 1 sun4c
```

*** **which** ***

Commande	Options	Paramètres
which		Nom de la commande

Description : Retourne le chemin complet d'accès à une commande.

Paramètres : le nom de la commande.

Exemples :

```
valjean ~ $ which ps
/usr/ucb/ps
```

*** **whereis** ***

Commande	Options	Paramètres
whereis		Nom de la commande

Description : Retourne le chemin complet d'accès à tous les répertoires contenant la commande, ainsi que les répertoires contenant les manuels de cette commande.

Paramètres : Le nom de la commande.

Exemples :



```
valjean ~ $ whereis ps
ps: /usr/bin/ps /usr/ucb/ps /usr/man/man1/ps.1 /usr/man/man1b/ps.1b
```

4.2 AUTRES COMMANDES

*** **DF** ***

Commande	Options	Paramètres
df		

Description : liste les disques montés sur la machine, en indiquant le pourcentage d'occupation.

*** **DU** ***

Commande	Options	Paramètres
du	-as	Nom de fichier ou répertoire

Description : affiche en kilo-octets la taille du fichier ou du répertoire.

Options :

- **s**: retourne uniquement la taille du répertoire.
- **a**: liste récursivement la taille de tout fichier et sous-répertoire à partir du noeud désigné.

Paramètres : nom du(des) fichier(s) ou répertoire(s). Sans paramètre, c'est le répertoire courant.

*** **OD** ***

Commande	Options	Paramètres
od	-dxc	Nom du fichier

Description : réalise un “dump” du fichier suivant le format spécifié.

Options :

- **d**: format décimal.

- **x**: format hexadécimal.
- **c**: format ASCII.

Paramètres : nom du fichier. Sans option le format est octal.

* * * **HEAD** * * *

Commande	Options	Paramètres
head	-n	fichier

Description : liste les n premières lignes du fichier.

Options :

- **n**: nombre de ligne affichées. Par défaut n=10.

Paramètres : nom du fichier.

* * * **TAIL** * * *

Commande	Options	Paramètres
tail	-n	Nom du fichier

Description : liste les n dernières lignes du fichier.

Options :

- **n**: nombre de lignes affichées. Par défaut n=10.

Paramètres : nom du fichier.

* * * **DIFF** * * *

Commande	Options	Paramètres
diff	-c	Fichier1 Fichier2

Description : compare les 2 fichiers et affiche les lignes différentes.

Options :

- **c**: affiche de part et d'autre de la ligne qui diffère les 3 lignes qui la précèdent et les 3 lignes qui la suivent. Les lignes supprimées sont marquées par un '-', les lignes rajoutées sont marquées par un '+', et les lignes modifiées par le signe '!'. La taille du contexte peut s'augmenter, par ex. -c10 indique 10 lignes avant et après.

Exemple :

```
diffessai.cessai.c.bak
diff -c applnew.c applold.c >appl.patches
```

* * * **UNIQ** * * *

Commande	Options	Paramètres
uniq	-u	source destination

Description : lit le fichier source, compare les lignes adjacentes, supprime les lignes similaires et recopie le résultat dans le fichier destination. Si le fichier destination est absent, la recopie se fera sur la sortie standard.

Options :

- **u**: seules les lignes uniques seront recopiées.

* * * **GREP, GGREP** * * *

Commande	Options	Paramètres
grep	-vinc	chaîne de caractères - nom de fichier

Description : affiche les lignes du fichier contenant la chaîne de caractères. La commande “ggrep” de GNU a des performances nettement meilleures que “grep”.

Options :

- **v**: seules les lignes ne contenant pas la chaîne seront affichées.
- **i**: la comparaison considère les majuscules et minuscules comme identiques.
- **n**: les numéros des lignes sont aussi affichés en début de ligne.
- **c**: grep retourne le nombre de comparaisons qui ont réussis.

Paramètres : si la chaîne comprend un ou plusieurs blanc, la mettre entre double quotes “ ”.

Exemple :

grep "From: " mbox (liste tous les champs From: de la boîte aux lettres)

* * * **SORT** * * *

Commande	Options	Paramètres
sort	-ru	Nom du fichier

Description : effectue le tri des lignes du fichier en les comparant caractère par caractère de gauche à droite suivant l’ordre dans la table ASCII (01...89...ABC...XYZ...ab...xyz...).

Options :

- **r**: tri par ordre inverse.
- **u**: élimine les lignes identiques.

* * * **FIND** * * *

Commande	Options	Paramètres
find “repertoire”	-name fichier -print	

Description : recherche récursive de fichier à partir du répertoire désigné, et satisfaisant aux options. Si on fait une recherche à partir du répertoire courant il faut spécifier le “.”.

Options :

- **name**: désigne le fichier à rechercher.

- **print**: affiche le chemin d'accès au fichier en cas de recherche réussie. Sans cette option même si le fichier existe, aucune impression n'aura lieu.

Exemple :

```
find . -name .login -print
find . -name "*.c" -print
```

* * * **AWK, GAWK** * * *

Commande	Options	Paramètres
awk	-F'...' 'commande'	fichier

Description : cette commande utilisée ici de façon élémentaire est en fait un véritable langage de programmation (Livre de ref: *the AWK programming language* de *Aho-Kernighan-Weinberger*). Elle permet d'extraire des informations d'un fichier. Le fichier est découpé en lignes, ces dernières étant découpées en champs, puis traité en fonction de la *commande* demandée. La commande "gawk" de GNU est plus conforme à la norme POSIX et est plus performante.

Options :

- **F'...'** permet de redéfinir le séparateur de champ, par défaut c'est le blanc.
- **'/.../{print \$n}'** imprime le champ *n* de chaque ligne comprenant la chaîne entre slashes. Si l'on omet **/.../** toutes les lignes sont traitées.

Paramètres : fichier à traiter. En cas d'omission on traite l'entrée standard.

Exemple :

```
df | awk -F'/' '{print $3}'
```

Remarques : est couramment employée comme filtre à la sortie d'un **pipe**.

* * * **TALK, YTALK** * * *

Commande	Options	Paramètres
talk		Destination

Description : permet de converser avec un utilisateur sur une machine distante.

Paramètres : la destination a la syntaxe suivante :

user@machine user étant le nom de l'utilisateur, machine le nom de la machine distante.

* * * **TAR** * * *

Commande	Options	Paramètres
tar	-cxtvpf	archive répertoire ou fichiers

Description : cette commande permet d'archiver ou de désarchiver une arborescence ou un groupe de fichiers. L'archive peut-être soit un périphérique (cartouche magnétique ou disquette), soit un fichier archive suffixé par la chaîne de caractères ".tar".

Options :

- **c** : indique que l'on va créer l'archive (create).
- **x** : indique que l'on va restaurer l'archive (extract).
- **t** : indique que l'on va lister le contenu de l'archive (type).
- **v** : mode verbeux, les caractéristiques de chaque fichier archivé (droits d'accès, propriétaire, groupe, taille, date,...) sont listées (verbose).
- **p** : préserve les dates et droits d'accès originels lors d'une restauration.
- **f** : indique que le nom du fichier qui suit désigne le nom de l'archive (archive.tar).

Paramètres : nom du ou des répertoires ou liste des fichiers. En restauration, si les paramètres sont absents toute l'archive est restaurée, sinon ils représentent les noms de fichiers à restaurer.

Exemple :

```
tar cvf tp.tar tp (sauvegarde du répertoire tp)
tar tvf tp.tar (listage du contenu de l'archive tp.tar)
tar xvf tp.tar (restauration l'archive tp.tar)
```

* * * COMPRESS, UNCOMPRESS * * *

Commande	Options	Paramètres
compress	-cfv	fichier
uncompress	-cv	fichier

Description : cette commande permet de compresser ou de décompresser un fichier. Le fichier compressé est suffixé par ".Z". Si la commande est utilisée sans option, le fichier original est remplacé par son compressé pour une compression et réciproquement pour une décompression.

Options :

- **c** : le résultat de la compression ou de la décompression est produit sur la sortie standard sans modifier le fichier origine. Cette option est pratique lorsque compress/uncompress est utilisé comme un filtre.
- **f** : Si un fichier compressé de même nom existe déjà, indique que l'on souhaite le réécraser (forçage).
- **v** : indique le taux de compression/décompression.

Paramètres : nom du fichier compressé ou décompressé.

Exemple :

```
compress tp.tar (compression de l'archive tp.tar en tp.tar.Z)
uncompress tp.tar.Z (décompression de l'archive tp.tar.Z en tp.tar)
```

* * * GZIP, GUNZIP * * *

Commande	Options	Paramètres
gzip	-cfdv9	fichier
gunzip	-cv	fichier

Description : cette commande est identique à la commande compress/uncompress. Cependant elle permet d'obtenir un gain de compression très largement supérieur. Le fichier compressé est suffixé par “.gz”. Si la commande est utilisée sans option, le fichier original est remplacé par son compressé pour une compression et réciproquement pour une décompression.

Options :

- **c** : le résultat de la compression ou de la décompression est produit sur la sortie standard sans modifier le fichier origine. Cette option est pratique lorsque gzip/gunzip est utilisé comme un filtre.
- **f** : si un fichier compressé de même nom existe déjà, indique que l'on souhaite le réécraser (forçage).
- **d** : indique d'effectuer une décompression (identique à gunzip).
- **v** : indique le taux de compression/décompression en fin de traitement.
- **9** : niveau de compression maximum (par défaut 5, minimum 1). Plus le niveau de compression est élevé, plus de traitement est long, inversement, si le niveau de compression est bas le traitement sera plus rapide.

Paramètres : nom du fichier compressé ou décompressé.

Exemple :

```
gzip tp.tar (compression de l'archive tp.tar en tp.tar.gz)
gunzip tp.tar.gz (décompression de l'archive tp.tar.gz en tp.tar)
```

* * * UUENCODE, UUDECODE * * *

Commande	Options	Paramètres
uuencode		fichier identification >fichier codé
uudecode		fichier

Description : la commande “uuencode” permet de coder un fichier binaire en un fichier éditable (au format ASCII) et ainsi peut être envoyé par “mail”. Inversement, la commande “uudecode” permet de restituer le fichier original à partir du fichier codé.

Paramètres : l'*identification* désigne généralement le nom du fichier qui a été codé. La commande “uuencode” produit le fichier codé en ASCII sur la sortie standard, c'est pour cela qu'il faut la rediriger dans un fichier final.

Exemple :

```
uuencode tp.tar.gz tp.tar.gz >tp.tar.uu (codage de l'archive tp.tar.gz en tp.tar.uu)
uudecode tp.tar.uu (décodage de l'archive tp.tar.uu en tp.tar.gz)
```

* * * MAIL, ELM * * *

Commande	Options	Paramètres
mail	-sfv	destinataire <fichier
elm	-sf	destinataire

Description : les commandes “mail” et “elm” permettent de lire, envoyer du courrier électronique. Il existe de nombreuses autres interfaces utilisateurs pour traiter la messagerie électronique (*mailx*, *mh*, *xhm*, *pine*, *mailtool*, *xmail*, *zmail*, *meuf*). L’interface de base reste la commande “mail” qui peut être utilisée de manière simple ou comme un filtre. La commande “elm” offre également une interface simple pour terminaux alphanumériques de type VT100/VT200.

Options :

- **s** : indique que la chaîne de caractères qui suit désigne le sujet, si celui-ci comporte des caractères ‘espace’ il est nécessaire de l’entourer par des doubles quotes.
- **v** : mode “*verbose*” qui permet de tracer le parcours du mail jusqu’à sa délivrance.
- **f** : indique que le nom du fichier qui suit désigne le “folder” (boîte aux lettres utilisateur ou dossier) que l’on veut lire. Si ce nom est absent, ce sera le fichier “mbox” qui sera pris par défaut. Si l’option est absente, la boîte aux lettres par défaut est “/usr/spool/mail/\$USER”.

Paramètres : indiquent, s’ils sont présents, le ou les destinataires séparés par un espace. En l’absence de paramètres, le courrier sera lu. Par défaut la commande “mail” lit le texte du message à envoyer sur l’entrée standard. La fin du texte doit être signalée par le caractère ‘.’ en première colonne suivi de <return>. Sinon, si le texte du message se trouve dans un fichier, il suffit d’utiliser le caractère de redirection ‘<’ pour lire le message à envoyer.

Exemple :

```
mail
mail -f mbox-untel
mail user1 user2
Subject: sujet
corps du message
.
mail -s "sujet" user1 user2 <message
```

Exercices :

- En utilisant la commande ‘du’, visualiser la taille totale de votre HOME directory, puis celle de chacun de vos sous-répertoires et fichiers.
 - Créer un fichier de quelques caractères et visualiser son contenu à l’aide de la commande ‘od’.
 - Après avoir visualisé les systèmes de fichiers montés sur votre machine (commande ‘df’), visualiser uniquement ceux qui appartiennent aux disques physiquement attachés à votre machine. Il sont désignés par ‘/dev/...’.
Pour cela on enchaînera la commande ‘df’ avec un PIPE sur la commande ‘grep’.
 - Créer un fichier dans le style suivant~:
ligne1
ligne2
ligne3
ligne4
 - On se propose de réaliser une permutation sur ce fichier, c’est à dire que la dernière ligne doit passer en tête.
Pour cela utiliser un fichier temporaire, les commandes ‘tail’ et ‘head’ ainsi que les opérateurs de redirection ‘>’ et ‘>>’.
 - On veut récupérer les noms de tous les propriétaires ayant des processus sur votre machine. On désire obtenir une liste par ordre alphabétique, chaque nom apparaîtra une seule fois.
Dans un premier temps on réalisera les différentes étapes en récupérant la sortie d’une commande dans un fichier, qui sera utilisé pour la commande suivante.
-
- Puis après mise au point, on pourra enchaîner nos commandes avec des PIPES de façon à réaliser l’opération en une seule ligne.
Commandes à utiliser~: ‘ps’, ‘awk’, ‘sort’, ‘uniq’, ‘grep’.
 - Si vous désirez communiquer avec une personne sur une autre machine utilisez la commande ‘talk’.

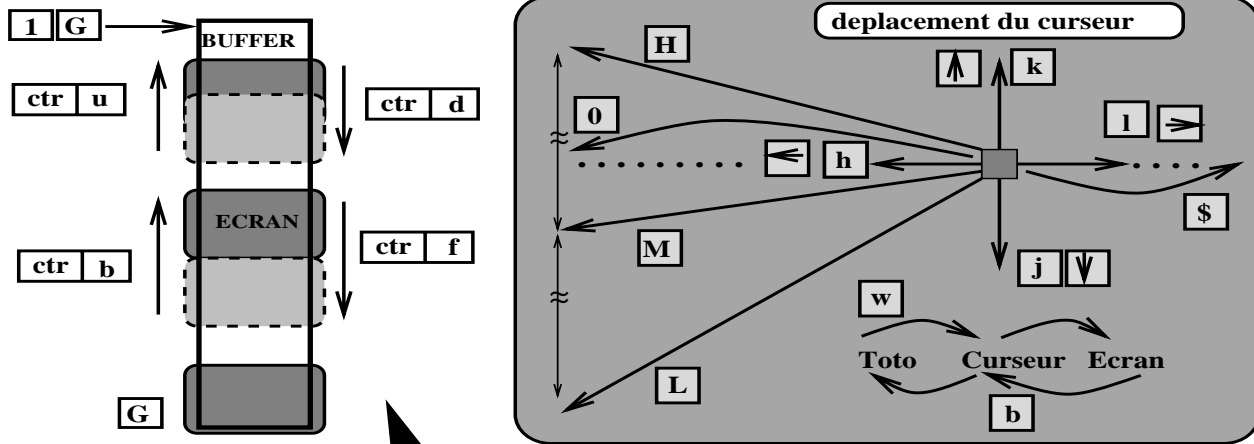
4.3 RÉSUMÉ DES COMMANDES VI

RÉSUMÉ DES COMMANDES VI

Dans la suite du texte :

- La notation $\hat{\langle \text{caractère} \rangle}$ signifie *touche-control* $\langle \text{caractère} \rangle$
- L'expression $\langle n \rangle$ est un entier, facteur de répétition optionnel.

Appel	
vi "nom de fichier"	
Sortie	
Avec sauvegarde	:wq ou :x
Sans sauvegarde	buffer non modifié :q
	buffer modifié :q!
Recherche	
Chaine de caractères	/ <i>< chaine ></i>
répétition	n ou N
Substitution	
mot	cw <i>< mot ></i> ESC
caractère	r <i>< car ></i>
Suppression	
Caractère	<i>< n ></i> x
Mots	<i>< n ></i> dw
Lignes	<i>< n ></i> dd
Récupération	
Après curseur	p
Avant curseur	P
Mémorisation	
Mot	<i>< n ></i> yw
Ligne	<i>< n ></i> yy ou <i>< n ></i> Y
Divers	
Rafraîchissement	\hat{l} (ctrl-l)
Inclure un fichier après le curseur	:r <i>< fichier ></i>
Exécution d'une commande shell	! <i>< commande ></i>
Annulation de la dernière commande	u



ATTENTION :

Lorsque vous etes en mode INSERTION , (fig ci-contre) pour revenir au mode commande (figs ci-dessus) taper la touche ESCAPE .

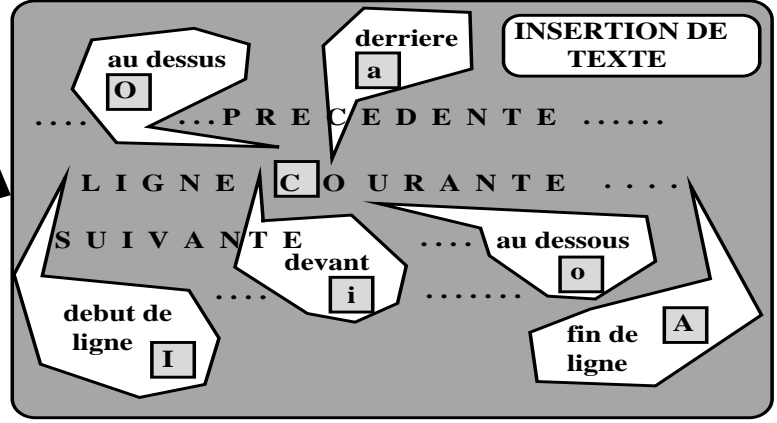


FIG. 4.1 - Commande VI

Index

Symbols

!, 34
!!, 34
↓, 34
←, 34
→, 34
↑, 34
*, 29
<, 28
>, 28
>>, 28
[...], 30
\ backslash, 30
| tube, 29
. , 9
.., 9
.profile, 31, 33, 34
.shenv, 35
.xinitrc, 35
.zshrc, 33
/ slash, 4, 9
/usr/local/startup, 34
;, 28
?, 30
\$, 31
~ Tilde, 28

A

alias, 33
arborescence, 4
arborescence locale, 6
arborescence reseau, 8
awk, 46

B

background, 29
bin, 6
Bourne shell, 30
BSD, 3

C

cal, 40
cat, 11

cd, 10
chmod, 13
compress, 47
control \, 19
control C, 19
control D, 19
cp, 12

D

daemon, 20
date, 40
descripteur de fichier, 17
dev, 6
df, 43
diff, 44
du, 43

E

echo, 41
elm, 49
environnement ENST, 34
etc, 6
exec, 16
export, 31

F

fichier droits d'accès, 13
fichier: commandes, 11
fichier: organisation, 5
file, 14
file de messages, 19
find, 45
fork, 16

G

gawk, 46
ggrep, 45
GID, 26
grep, 45
gunzip, 48
gzip, 48

H

head, 44

history, 34

I

id, 38

inode I-list, 5

install-startup, 34

interpréteur de commande, 25

IPC System V, 19

K

kill, 20

L

ligne de commande shell, 26

ln, 13

login, 26

logname, 37

ls, 10

M

mémoire partagée, 19

mail, 49

man, 9

metacaractère, 29

mkdir, 10

more, 12

mv, 12

N

NFS, 3, 8

NIS, 3

O

od, 43

opt, 6

P

password, 26

PATH, 9, 31

pid, 21

processus, 16

prompt, 26

ps BSD, 22

ps System V, 23

PS1, 31

pwd, 9

R

read, 30

redirection, 17, 28

rlogin, 39

rm, 13

rmdir, 10

root, 4

S

sémaphore, 19

shell, 25

signaux, 19, 31

sleep, 41

socket, 19

sort, 45

startx, 35

stderr, 17

stdin, 17

stdout, 17

super-block, 5

System V, 3

T

tab, 34

tail, 44

talk, 46

tar, 46

telnet, 40

TERM, 31

tmp, 6

trap, 31

tty, 41

tube, 18

tube nomme, 18

U

UID, 26

unalias, 33

uname, 42

uncompress, 47

uniq, 44

Unix, 3

unset, 31

usr/include, 6

usr/lib, 6

usr/local/bin, 6

uudecode, 48

uuencode, 48

V

var, 6

variable définition, 30

variable shell, 30

variable substitution, 31

W

w, 38

wait, 16
whereis, 43
which, 42
who [am i], 37

Y

ytalk, 46

Z

zsh, 33