

SUPPORT
UNIX UTILISATEUR
Thierry GRANDADAM

TABLE DE MATIERES

1. GENERALITES	1
1.1. La notion de système d'exploitation	1
1.2. Historique du système UNIX	1
1.3. Caractéristiques du système	1
1.3.1. Portabilité du système	1
1.3.2. Description générale du shell	2
1.4. Le système de fichier	3
1.5. Le travail en multipostes	4
1.6. Les traitements multitâches	5
2. Démarrage & ACCES AU système	6
2.1. Démarrage	6
2.2. Connexion au système et déconnexion	6
2.3. Obtention du nom de connexion	7
2.4. Changement de mot de passe utilisateur	7
2.5. Obtention du nom d'utilisateur	8
2.6. Obtention de l'état des connexions	8
2.7. Obtention du nom de la console	9
2.8. Identifications des groupes et utilisateurs	9
3. GESTION DU TEMPS	10
3.1. Affichage ou modification de la date	10
3.2. Affichage du calendrier	11
3.3. Affichage des notes du jour	11
4. GESTION DE FICHIERS	12
4.1. Affichage du catalogue	12
4.2. Concaténation et affichage de fichiers	13
4.3. Dénombrement des mots lignes et caractères d'un fichier	14
4.4. Comparaison de deux fichiers	14
4.5. Détermination du type de fichier	14
4.6. Copie de fichiers	15
4.7. Changement de permission d'accès	15
4.8. Changement de propriétaire, de groupe d'un fichier	16

4.9.	Changement de nom ou déplacement de fichiers	16
4.10.	Suppression de fichiers	17
4.11.	Recherche d'occurrence dans un fichier	17
4.12.	Compactage/décompactage de fichiers	18
4.13.	Archivage et sauvegarde	18
4.14.	Recherche de fichiers	19
5.	MANIPULATION DES REPERTOIRES	20
5.1.	Création de répertoire	20
5.2.	Suppression de répertoire	20
5.3.	Affichage du chemin courant	20
5.4.	Changement de répertoire	20
6.	LES REDIRECTIONS	21
6.1.	Introduction	21
6.2.	Redirection de sortie	21
6.3.	Redirection d'entrée	22
6.4.	Redirection de l'erreur	22
6.5.	Création d'un tube	22
6.6.	Enchaînement simple	22
6.7.	Enchaînement complexe	23
6.8.	Ordre de priorité	23
6.9.	Les filtres	23
7.	LES IMPRESSIONS	24
7.1.	Impression de fichier	24
7.2.	Annulation d'impression	24
7.3.	Activation/Déactivation d'une file d'attente	25
7.4.	Obtention de l'état de l'imprimante	25
8.	MESSAGERIE	26
8.1.	Envoyer du courrier	26
8.2.	Conversation en temps réel	26
8.3.	Autorisation de réception de message	27
9.	LES PROCESSUS	28
9.1.	Introduction	28
9.2.	Lancement d'un processus en tâche de fond	28
9.3.	Destruction d'un processus	29

9.4. Programmation d'un processus	29
10. L'EDITEUR VI	30
10.1. Introduction	30
10.2. Lancement et fin de vi	30
10.3. Commande d'insertion caractères/lignes	31
10.4. Touches de déplacement	31
10.5. Commandes de suppression de texte	31
10.6. Insertion des tampons	32
10.7. Commandes d'annulation/répétition	32
10.8. Recherche et remplacement d'occurrences	32
10.9. Commandes d'entrées/sorties	32
10.10. Commandes diverses	32
11. LE BOURNE SHELL	33
11.1. Lancement	33
11.2. Caractères spéciaux	33
11.2.1. Génération de noms de fichiers	33
11.3. Quotation des chaînes et des commandes	35
11.4. Programmation en Bourne shell	35
11.4.1. Les Variables	36
11.4.2. Les variables d'environnement	36
12. LE C SHELL	37
12.1. Lancement du C shell	37
12.2. Mécanismes d'historique	37
12.3. Création et suppression d'alias	37
12.3.1. Création d'un alias	37
12.3.2. Suppression d'un alias	38
12.4. Programmation en C shell	38
12.5. Les variables	38
12.5.1. Les variables ordinaires	38
12.5.2. Les variables prédéfinies	39
12.5.3. Les variables d'environnement	40

1. GENERALITES

1.1. La notion de système d'exploitation

Le système d'exploitation est le programme de base de la machine; il a essentiellement deux rôles :

- Gérer le matériel (périphériques, mémoire...).
- Offrir au programmeur une vision normalisée de la machine, indépendante de l'électronique de tel ou tel constructeur.

1.2. Historique du système UNIX

Issu d'une recherche commune aux laboratoires Bell et à la General Electric, le système MULTICS devait être capable de gérer une ville entière équipée de terminaux.

Ce système ne fonctionna jamais et un des informaticiens des laboratoires Bell, Ken Thompson conçut un système qu'il nomma par dérision UNIX. Il n'imaginait pas alors l'avenir de son système ! La première édition date de 1971, où furent mis en place la notion de système de fichiers, la gestion des processus et divers utilitaires.

En 1972, la seconde édition sur l'ordinateur PDP11 apporta les tubes de communication (pipes). Ken Thompson travailla en collaboration avec Ritchie, l'un des inventeurs du langage C (avec kernighan), pour réécrire UNIX dans ce langage dans sa quasi-totalité, hormis quelques lignes écrites en assembleur.

A partir de 1975, des licences furent accordées et il était possible de disposer des sources d'UNIX.

Deux grandes familles d'UNIX ont commencé à se former à partir de cette date pour arriver en 1987 à UNIX SYSTEM V release 3 d'une part et BERKELEY 5.0 d'autre part. Ces deux grandes lignes convergent aujourd'hui pour se rejoindre dans la version 4 de SYSTEM V.

1.3. Caractéristiques du système

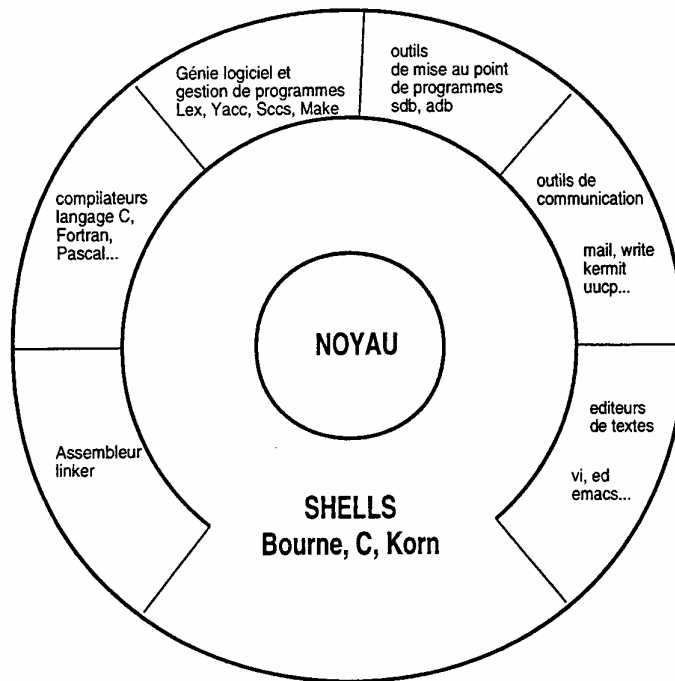
1.3.1. Portabilité du système

L'un des grands points forts du système Unix est sa possibilité de fonctionner sur n'importe quelle plate-forme matérielle, du supercalculateur au micro PC AT car il est écrit à plus de 90% en langage C, qui est un langage "portable". Seules quelques routines de base sont écrites en assembleur.

La plupart des informations à l'intérieur du système sont stockées sous forme de fichiers textes aisément modifiables.

UNIX fonctionne aujourd'hui sur un très large éventail de machines et processeurs Motorola, Intel, machine Risc...

ARCHITECTURE DU SYSTEME UNIX



1.3.2. Description générale du shell

Une commande est une séquence de mots séparés par des espaces blancs. Le premier mot d'une ligne de commande est le nom de la commande. Les mots suivants représentent les arguments de la commande.

Un programme chargé d'interpréter les commandes tapées par l'utilisateur se nomme un shell.

Le shell dispose d'un langage assez complet. Il autorise les manipulations de variables, l'utilisation de structures alternatives et répétitives. C'est à partir du shell que sont lancés tous les programmes servant aux utilisateurs.

Pour que le shell exécute une commande, il faut terminer la frappe de la commande par la touche d'avance ligne, matérialisée sur la plupart des claviers par "Entrée".

Une commande en cours de frappe peut être complètement annulée par la frappe sur la touche "Suppr".

Par défaut, le shell lit les commandes à partir d'un flux nommé stdin (0) correspondant au clavier, il écrit sur la sortie standard stdout (1) ou sur la sortie d'erreur stderr (2) (écran).

Ces trois flux peuvent être déviés vers un fichier disque ou vers un périphérique. C'est ce que l'on nomme une redirection d'entrée ou de sortie.

Le shell dispose en outre de caractères spéciaux utilisés pour les substitutions dans les commandes

Pour stopper provisoirement un défilement écran, vous pouvez utiliser la touche "Ctrl" et "S" puis reprendre le défilement par "Ctrl" et "q".

ATTENTION ! Le shell distingue les majuscules des majuscules et ne reconnaîtra pas une commande Si elle n'est pas frappée de manière adéquate. La plupart des commandes sont en minuscules.

Il existe plusieurs shell :

- Le shell standard est le BOURNE SHELL du nom de son concepteur. Son signe distinctif (prompt) est le dollar \$
- C SHELL est un interpréteur plus puissant, dont la syntaxe est inspirée du langage C. Son signe distinctif est le pourcentage %.
- Le KORN SHELL est également assez répandu. Son signe distinctif est le pourcentage %

1.4. Le système de fichier

Un fichier est composé d'une suite d'octets qui, du point de vue de l'utilisateur, apparaît continue, mais qui est en fait répartie sur des blocs qui ne sont pas forcément adjacents.

UNIX sait retrouver les blocs composant un fichier grâce à des modes (nœuds).

Il existe trois types de fichiers :

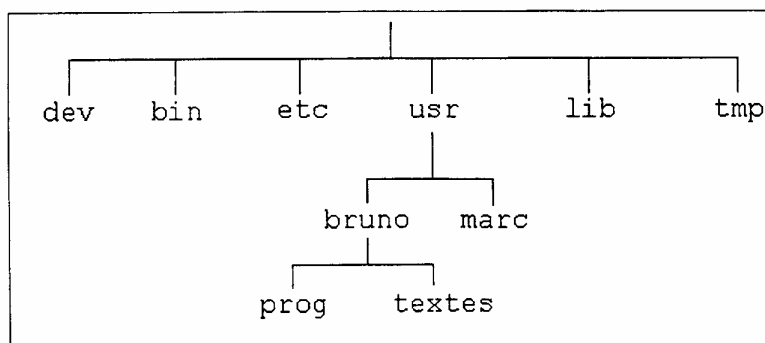
- fichiers ordinaires (données, programmes).
- fichiers répertoires.
- fichiers spéciaux (périphériques, mémoires, tubes, sémaphores...).

Unix possède un système de gestion des fichiers élaboré. Tout est fichier sur Unix, y compris (et surtout !) les périphériques.

Organisé en arborescence de répertoires, le système de fichiers est un arbre où peut être rajoutée (montée) une branche, correspondant par exemple, à une disquette ou à un autre disque dur.

Les répertoires principaux sont :

- / racine.
- /dev répertoire des fichiers spéciaux.
- /bin répertoire des commandes et utilitaires.
- /lib répertoire des librairies.
- /etc répertoire des fichiers et utilitaires d'administration.
- /tmp répertoire des fichiers temporaires.
- /usr répertoire des utilisateurs.



Les protections sont rigoureuses et pour lire, écrire, exécuter un fichier ou se placer dans un répertoire, il faut impérativement disposer de l'autorisation adéquate.

Un seul utilisateur a accès à toutes les informations : le superviseur, encore nommé administrateur système ou super-utilisateur. Il se connecte au système sous l'appellation root. Son répertoire d'accueil est la racine (/) d'où le nom de connexion de root.

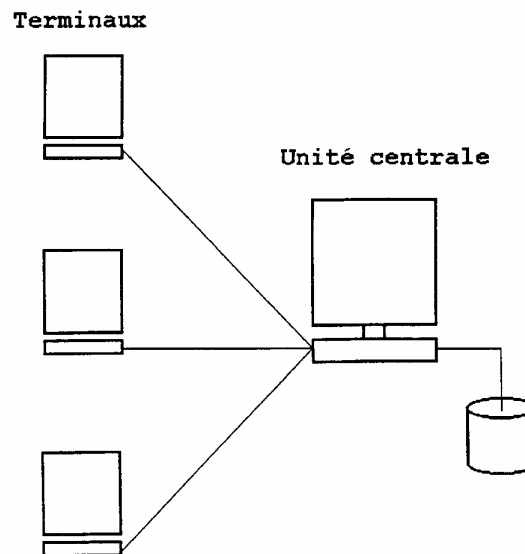
C'est lui qui a en charge la maintenance et l'organisation du système. Il crée les login (noms de connexion) pour les utilisateurs et leur accorde les privilèges d'accès.

On ne doit se connecter ou passer en mode superviseur que lorsque cela est réellement nécessaire. D'ailleurs, en règle générale, la personne chargée de maintenir le système aura également un compte pour se connecter en mode utilisateur standard. Ceci restreint les risques de manipulations erronées, les pouvoirs du superviseur étant sans limites!

1.5. Le travail en multipostes

Unix appartient à la famille des systèmes multipostes: il est possible de travailler simultanément à plusieurs sur la même unité centrale, reliés par un terminal (écran + clavier) ou par un micro-ordinateur émulant (simulant) un terminal.

Exemple d'un schéma multipostes



L'ordinateur partage son temps entre les utilisateurs très rapidement, donnant l'impression à chacun qu'il dispose de l'ordinateur pour lui tout seul. Seul le nombre de terminaux et la lourdeur des tâches peut affecter la performance du système. C'est ce que l'on nomme le temps partagé

Du fait qu'il s'agit d'un système où l'on travaille à plusieurs, il convient d'observer une règle d'extrême PRUDENCE, surtout si l'on dispose d'autorisations importantes quant à l'accès à certains fichiers stratégiques (c'est le cas notamment du superviseur).

1.6. Les traitements multitâches

Unix est en outre un système véritablement multitâches : il est possible de lancer un processus en tâche de fond tout en continuant à travailler sur autre chose. Vous pouvez ainsi éditer un long document ou trier un fichier tout en continuant un autre travail.

Un processus est différent d'un programme (vous pouvez assimiler un programme à une recette de cuisine et un processus à une personne en train d'exécuter la recette de cuisine !).

Un processus dispose d'un espace d'adressage (code+données+pile) et d'un environnement (état des registres, identificateur de processus ...).

2. DEMARRAGE & ACCES AU SYSTEME

2.1. Démarrage

Après la mise sous tension du matériel et des périphériques, a lieu le bootstrap (chargement et exécution).

En principe, le système demande ensuite la date et l'heure, qu'il convient de renseigner correctement.

Il effectue alors le montage des disques, puis la vérification des fichiers.

Viennent ensuite le démarrage en mono-utilisateur, puis le passage en multi-utilisateurs.

Toutes ces étapes sont appelées les run-levels (niveaux d'exécution).

On ne doit jamais éteindre une machine UNIX sans avoir effectué les opérations adéquates de terminaison (ces opérations sont réservées au superviseur). En effet, certaines informations en mémoire ne sont écrites que périodiquement sur le disque (points de synchronisation). Si une coupure de courant intervient inopinément, le système peut donc se trouver désorganisé.

L'extinction d'un terminal est par contre sans danger pour le système.

2.2. Connexion au système et déconnexion

Un utilisateur est repéré par:

- un nom
- un mot de passe (facultatif Si le superviseur en décide ainsi)
- un numéro d'identification (id)
- un nom de groupe de travail
- un numéro de groupe (gid)
- un répertoire d'accueil (/usr/nom)
- facultativement, un ou plusieurs programmes à lancer.

L'utilisateur doit fournir son nom lors du login :

```
ENISERV
Bienvenue sous UNIX SCO System V

ENISERV!login: _
```

Il est ensuite tenu de fournir son mot de passe qui n'apparaît pas sur l'écran lors de la frappe. Si le mot de passe est correct, l'utilisateur est alors connecté au système.

Le nombre de tentatives d'entrée du mot de passe est fixé par le superviseur:

Il n'existe pas de lien entre la console physique et l'utilisateur : tout utilisateur peut se connecter sur n'importe quel terminal. Il peut même se connecter plusieurs fois sous le même nom sur différentes consoles.

Pour se déconnecter, utilisez la séquence de touche ctrl et d ou l'instruction `logout` ou bien encore `exit`.

Il est important de ne pas oublier de se déconnecter, notamment dans le cas d'une extinction du système. Dans le cas contraire, certains fichiers peuvent être endommagés.

2.3. Obtention du nom de connexion

Cette commande permet d'obtenir le nom de connexion. Elle est utilisée sans paramètres.

Syntaxe **logname**

L'utilisateur Bruno demande l'affichage de son nom de connexion:

```
$ logname
bruno
$
```

2.4. Changement de mot de passe utilisateur

Cette commande permet à un utilisateur de changer régulièrement son mot de passe. Il doit alors fournir l'ancien mot de passe et saisir deux fois le nouveau (pour des raisons de sécurité), sans que celui-ci apparaisse à l'écran.

Le superviseur peut imposer aux utilisateurs de changer régulièrement de mot de passe. Le système en fera alors la demande à la connexion.

Syntaxe **passwd**

L'utilisateur bruno change de mot de passe:

```
$ passwd
setting password for user bruno
old password
new password
Re-enter password
$
```

On ne doit sous aucun prétexte oublier son mot de passe car il est impossible de le retrouver. Si un utilisateur oublie son mot de passe, le superviseur devra lui en attribuer un nouveau. Si le superviseur oublie son mot de passe il faut RÉINSTALLER LE SYSTÈME!.

2.5. Obtention du nom d'utilisateur

Cette forme d'appel de la commande `who` permet d'obtenir son nom de connexion, le nom de la console la date et l'heure de connexion.

Syntaxe `who am i`

L'utilisateur bruno demande des informations sur sa connexion:

```
$ who am i
bruno      tty01  Nov 02 08:45
$
```

2.6. Obtention de l'état des connexions

Indiquer qui utilise le système.

Syntaxe `who -uHqbl`

Paramètres

- u Affichage du temps écoulé depuis la dernière opération et du numéro de processus Shell.
- H Affichage des entêtes de colonne.
- q `who` rapide, nom des users + nombre d'utilisateurs.
- b Date et heure du dernier reboot.
- l Lignes en attente de session.

Demande d'affichage des utilisateurs connectés:

```
$ who
bruno          tty01  Jan 28 08:00
marc           tty02  Jan 28 08:45
$
```

2.7. Obtention du nom de la console

Obtenir le nom de la console en cours d'utilisation.

Syntaxe **tty**

```
$ tty
/dev/tty08
$
```

2.8. Identifications des groupes et utilisateurs

Fournir le numéro d'utilisateur et de groupe (uid et gid).

Syntaxe **id**

L'utilisateur bruno demande l'affichage de son numéro d'utilisateur et de groupe:

```
$ id
uid=201(bruno) gid=100(interne)
$
```

3. GESTION DU TEMPS

3.1. Affichage ou modification de la date

Afficher ou modifier la date système.

Syntaxe **date** **mmddhhmmyy** **+format**

Les éléments suivants peuvent être placés dans le format afin de contrôler la présentation. Le format sera délimité par des quotes (').

- %D Date sous la forme JJ/MM/AA.
- %T Heure sous la forme HH:MM:SS.
- %m Mois de 01 à 12.
- %d Jour de 01 à 31.
- %y Année de 00 à 99.
- %H Heure de 00 à 23.
- %M Minute de 00 à 59.
- %S Seconde de 00 à 59.
- %j Jour de l'année de 001 à 366.
- %w Jour de la semaine de 0 à 6 (0=dimanche)
- %a Abréviation du jour (dim à sam).
- %h Abréviation des mois (Jan à dec).
- %r Heure en notation AM/PM.

Affichage de la date courante puis changement de la date système et affichage de la date formatée:

```
$ date
Jeu 28 Nov 08:25:23:  gmtd 1991
$ date 1130093291
Sam 30 Nov 09:32:00 gmtd 1991
$ date +%a %d %h %r
Jeu 28 Nov 08:33:18 AM
```

| *Seul l'administrateur peut changer la date système.*

3.2. Affichage du calendrier

Imprimer le calendrier.

Syntaxe cal mois année

La commande cal sans aucun paramètre affiche le calendrier du mois précédent, du mois courant et du mois suivant.

- mois Donne le calendrier du mois précisé.
- année Fournit le calendrier des douze mois de l'année précisée.

Affichage du calendrier de Novembre:

```
$ cal Nov
November 1991
S M Tu W Th F S
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
$
```

3.3. Affichage des notes du jour

Il est possible d'enregistrer une sorte d'agenda sous la forme d'un fichier texte dans un répertoire utilisateur. Ce fichier doit contenir des lignes au format: mois jour message (elles ne sont pas obligatoirement dans l'ordre chronologique).

Le fichier en question doit se nommer calendar. Lors de la frappe de la commande calendar, seront affichées les lignes de la date courante et du lendemain.

Syntaxe calendar

La date du jour étant le 28 et le fichier nommé calendar comportant les lignes suivantes:

Novembre 28 penser à appeler Sylvie

Novembre 29 écrire à monsieur Martin

```
$ calendar
November 28 penser à appeler Sylvie
November 29 écrire à monsieur Martin
```

4. GESTION DE FICHIERS

4.1. Affichage du catalogue

Affichage du catalogue du répertoire.

Syntaxe **ls -AaRdcxlnogrtspfiu**

Paramètres

- A Affiche toutes les entrées y compris celles commençant par . sauf le répertoire courant.
- a Identique à A mais affiche le répertoire courant (.) et la branche supérieure (..)
- R Listing récursif de tous les sous-répertoires rencontrés.
- d Affiche les informations sur un ou plusieurs répertoires.
- C Sortie multi-colonnes triée par colonne.
- x Sortie multi-colonnes triée par ligne.
- m Sortie sous forme de flot (les noms sont séparés par la virgule)
- l Affichage au format long.
- n Identique à l mais affiche uid et gid.
- o Identique à l mais sans le groupe.
- g Identique à l mais sans le propriétaire.
- r Inversion de l'ordre de tri.
- t Tri par date de dernière modification au lieu du nom.
- s Taille en bloc.
- p Ajoute un slash (l) derrière chaque répertoire.
- f Identique à p mais ajoute un astérisque derrière les fichiers exécutables.
- i Affichage du numéro d'inode.
- u Tri par date de dernier accès.

Affichage du catalogue au format long:

Le premier caractère représente le type de fichier:

- Fichier ordinaire.
- d Répertoire.
- c Périphérique caractère.
- b Périphérique bloc.

Viennent ensuite l'état de protection, pour l'utilisateur, le groupe et les autres

- Permission non accordée.
- r Accès autorisé en lecture.

w Accès autorisé en écriture.
x Accès autorisé en exécution.

La seconde colonne indique le nombre de liens pour un fichier.
S'il s'agit d'un répertoire, le nombre de liens correspond au nombre de sous-répertoires.
Si le répertoire ne contient aucun sous-répertoire, le nombre de liens vaut 2 (pour . et .., le répertoire courant et le répertoire père)

La troisième et la quatrième colonne fournissent respectivement le nom du propriétaire et le nom du groupe.

La colonne suivante donne la taille en octets du fichier.
La sixième colonne indique la date et l'heure de la dernière modification.

La septième et dernière colonne fournit le nom du fichier.

4.2. Concaténation et affichage de fichiers

Concaténation et affichage de fichiers. Suivie de plusieurs noms de fichiers, cette commande les affiche les uns à la suite des autres. Elle est très utilisée en redirection de sortie pour rassembler plusieurs textes dans un nouveau fichier.

Syntaxe cat fichier1 fichier2

Affichage des fichiers texte i et texte2. Concaténation dans un fichier texte3 par redirection de sortie:

4.3. Dénombrement des mots lignes et caractères d'un fichier

La commande `wc` permet de dénombrer les lignes, mots et caractères d'un texte.

Syntaxe `wc -wcl fichier`

Paramètres

- l Affichage uniquement du nombre de lignes.
- w Affichage du nombre de mots.
- c Affichage du nombre de caractères.

Dénombrement des lignes du fichier texte 1:

```
$ wc -l texte1
2 texte1
```

4.4. Comparaison de deux fichiers

Comparer deux fichiers.

Syntaxe `cmp fichier1 fichier2`

Comparaison des fichiers texte1 et texte2 (ils sont différents) puis comparaison du fichier texte1 avec lui-même (égalité):

```
$ cmp texte1 texte2
texte1 texte2 differ: char 18, line 1
```

4.5. Détermination du type de fichier

Déterminer le type de fichier.

Syntaxe `file fichier`

Affichage des types de tous les fichiers du répertoire courant:

4.6. Copie de fichiers

Copie de fichiers.

Syntaxe **cp fichier fichier**
 cp fichier répertoire

Si la destination est le même répertoire que celui d'origine, le nom d'arrivée doit être différent.

Copie le fichier texte1 sous le nom texte4:

```
$ cp texte1 texte2
$
```

4.7. Changement de permission d'accès

Cette commande permet de changer les permissions d'accès à un fichier ou répertoire. Sont distingués le propriétaire, le groupe de travail, et les autres. Les permissions sont au nombre de trois; lecture, écriture, exécution.

Pour un répertoire, la permission de lecture correspond à la possibilité de lister le contenu du répertoire. La permission d'écriture autorise la création ou la suppression de fichiers dans le répertoire. La permission d'exécution autorise le passage (placement) dans le répertoire.

Ces permissions ne sont pas transmises aux éventuels sous-répertoires.

Seul le propriétaire d'un fichier ou le superviseur peut changer les permissions d'un fichier.

Syntaxe **chmod ugoa +- permission fichier**

Paramètres

u	Propriétaire.
G	Groupe.
o	Les autres.
a	Tout le monde (par défaut).
+	Ajout de la permission.
-	Retrait de la permission.
r	Lecture.
w	Ecriture.
x	Exécution.

La permission peut également être fournie en absolu sous la forme d'un nombre.

0400 autorisation de lecture pour le propriétaire.

0200 autorisation d'écriture pour le propriétaire.

0100 autorisation d'exécution pour le propriétaire.

0040 autorisation de lecture pour le groupe.

0020 autorisation d'écriture pour le groupe.

0010 autorisation d'exécution pour le groupe.

0004 autorisation de lecture pour les autres.

0002 autorisation d'écriture pour les autres.

0001 autorisation d'exécution pour les autres.

Ces chiffres peuvent être combinés (le OU logique) pour indiquer les permissions d'un fichier.

Autorisation pour le groupe de lire le fichier monprog:

```
$ chmod g+r monprog  
$
```

Accès en lecture seulement pour tout le monde sur le fichier monprog:

```
$ chmod a+r monprog  
$
```

Le fichier monprog devient exécutable par son propriétaire:

```
$ chmod u+x monprog  
$
```

Le fichier monprog est accessible en lecture / écriture / exécution pour tout le monde:

```
$ chmod 0777 monprog  
$
```

4.8. Changement de propriétaire, de groupe d'un fichier

La commande **chown** ou **chgrp** permet de changer le propriétaire d'un fichier ou son groupe.

Elle peut être exécutée seulement par le propriétaire ou le superviseur. La donation est irréversible et seul le nouveau propriétaire peut vous redonner la propriété!

Syntaxe chown propriétaire fichier
 chgrp groupe fichier

Marc devient le propriétaire du fichier texte 4:

```
$ chown marc texte4  
$
```

4.9. Changement de nom ou déplacement de fichiers

Déplacement ou changement de nom de fichiers ou répertoires.

Syntaxe mv fichier1 fichier2
 mv répertoire1 répertoire2
 mv fichier répertoire

Changement de nom du fichier texte1 en texte5:

```
$ mv texte1 texte5  
$
```

Déplacement du fichier texte5 dans le répertoire /usr/tmp:

```
$ mv texte5 /usr/tmp
$
```

4.10. Suppression de fichiers

La commande **rm** efface les fichiers ou les répertoires.

Syntaxe `rm -fri fichier`

Paramètres

- f Pas de demande de confirmation pour un fichier dont on ne dispose pas de la propriété.
- r Effacement complet de répertoire (dangereux!)
- i Demande d'une confirmation de l'effacement

Suppression du fichier texte2 puis du fichier texte5:

```
$ rm texte2
$ rm /usr/tmp/texte5
```

4.11. Recherche d'occurrence dans un fichier

La commande **grep** recherche un texte dans un ou plusieurs fichiers.

Syntaxe `grep -vclhnsy expression fichier`

D'une manière générale, encadrez l'expression de simples quotes.

Paramètres

- v affiche toutes les lignes sans l'occurrence
- c affiche le nombre de lignes comportant l'occurrence.
- l affiche seulement le nom des fichiers comportant l'occurrence.
- h supprime le nom de fichier en début de ligne.
- n affiche le numéro de chaque ligne comportant l'occurrence.
- s suppression des messages d'erreur pour un fichier inexistant.
- y recherche sans distinction de majuscule et minuscule.

Trouvera ta1 ta2 ta3 dans n'importe quel fichier du répertoire courant:

```
$ grep 'ta[123]' *
$
```

4.12. Compactage/décompactage de fichiers

La commande pack et unpack vous permet de gagner de la place de l'ordre de 30 à 35%. Au nom de fichier sera ajoutée l'extension .z

Syntaxe pack -f fichiers
 unpack fichiers

Paramètres

-f Force le compactage de fichiers. Sert à indiquer que tout un répertoire doit être compacté, même si certains fichiers n'en bénéficieront pas.

4.13. Archivage et sauvegarde

La commande **tar** permet de faire une sauvegarde de fichiers sur bande magnétique.

Syntaxe tar crutxvw fichiers
Attention il n'y a pas de tiret pour les options

Paramètres

c Sauvegarde sur bande à partir du début de la bande (écrasement).
r Ajout enfin de bande
u Mise à jour des fichiers modifiés sur la bande sans écrasement.
t Affichage du contenu de la bande.
x Extraction depuis la bande.
v La commande tar affiche les traitements en cours (verbiage).
w Demande de confirmation pour chaque traitement.

Sauvegarde de tous les fichiers et répertoires avec ECRASEMENT de la bande.

4.14. Recherche de fichiers

La commande **find** recherche des fichiers.

Syntaxe find chemin expression

L'expression peut comporter :

-name nom	Recherche le fichier en fonction de son nom.
-user nom	Recherche sur l'appartenance.
-ctime n	Vrai si le fichier a été modifié ou créé n jours auparavant.
-print	Affiche le nom de fichier en cours de traitement.

Il est fortement conseillé de tester la commande find avec seulement -print pour vérifier que la commande portera sur les fichiers souhaités!

Affichage de tous les fichiers du répertoire courant commençant par la lettre t :

```
$ find . -name "t*" -print
./texte3
./texte4
```

5. MANIPULATION DES REPERTOIRES

5.1. Création de répertoire

La commande **mkdir** permet de créer un répertoire.

Syntaxe mkdir nomrépertoire

Création du répertoire monrep :

```
$ mkdir monrep
$
```

5.2. Suppression de répertoire

La commande **rmdir** permet la suppression d'un répertoire. Celui-ci doit être vide.

Syntaxe rmdir -p nomrépertoire

Paramètres

-p Suppression des répertoires parents (si vides).

Suppression du répertoire monrep :

```
$ rmdir monrep
$
```

5.3. Affichage du chemin courant

La commande **pwd** permet d'afficher le chemin courant .

Syntaxe pwd

5.4. Changement de répertoire

La commande **cd** permet de changer de répertoire courant. Si aucun paramètre n'est fourni, cette commande permet de retourner au répertoire utilisateur.

Le méta-caractère ~ correspond au répertoire utilisateur pour le C-shell.

Syntaxe cd répertoire

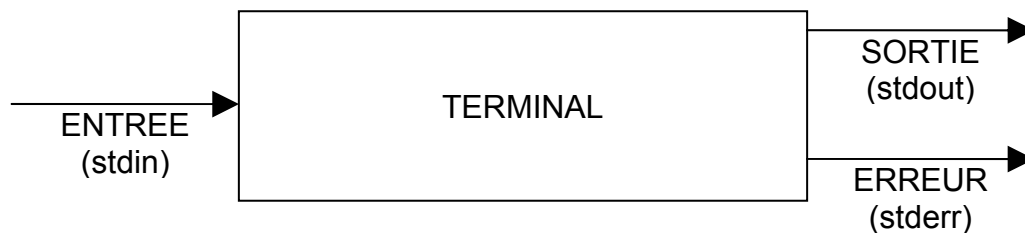
Accède au répertoire racine :

```
$ cd /
```

6. LES REDIRECTIONS

6.1. Introduction

Certaines commandes ne produisent pas de sorties à l'écran. Par exemple la commande **cp** crée une copie mais n'écrit rien sur le terminal sauf si il y a une erreur. Par contre la commande **who** font des sorties à l'écran. On peut résumer ceci par le schéma suivant :



6.2. Redirection de sortie

Une commande UNIX produit par défaut un éventuel affichage sur le fichier stdout, correspondant à la console.

Cette sortie peut être réorientée pour s'effectuer non pas à l'écran, mais vers un fichier (ordinaire ou spécial) afin de conserver une trace du résultat sous la forme d'un texte ou pour l'expédier vers un périphérique.

Si le fichier existe, il est détruit puis recréé.

Le signe > effectue une redirection vers un fichier en le recréant s'il existe.

Le signe >> envoie le résultat de la commande à la fin d'un fichier.

Si le résultat ne doit pas être imprimé ni stocké dans un fichier disque, il faut rediriger la commande vers /dev/null qui est un puits sans fond d'où rien ne ressort.

L'affichage du catalogue se fera dans un fichier appelé monfic :

```
$ ls > monfic
$
```

L'affichage du catalogue est ajouté à la fin du fichier monfic :

```
$ ls >> monfic
$
```

6.3. Redirection d'entrée

Toute commande Unix reçoit en principe ses instructions de stdin qui correspond au clavier. Cette entrée peut être réacheminée en provenance d'un fichier texte qui remplacera l'opérateur au clavier pour la commande.

Envoi d'un message à Marc. Le message n'est pas saisi au clavier mais a été tapé au préalable dans un fichier de nom 'montexte.txt' :

```
$ write marc < montexte.txt
$
```

Une redirection permet également d'exécuter sans intervention des commandes qui attendent en principe une confirmation de l'utilisateur. Elles recevront alors confirmation par le fichier placé en entrée.

6.4. Redirection de l'erreur

Avec le bourne shell vous pouvez rediriger la sortie stderr vers un fichier grâce au symbole **2>** ou **2>>** et **>&** en cshell.

```
$ cat f1 f2 2> erreurs
$ cat erreurs
```

6.5. Création d'un tube

Il est possible d'acheminer la sortie d'une commande comme entrée d'une autre commande à l'aide d'un pipe (tube). C'est un enchaînement de commande.

La commande d'impression lp est couramment employée à l'aide d'un tube.



```
$ ls /dev | more
$
```

6.6. Enchaînement simple

Vous pouvez enchaîner très simplement des commandes sur la même ligne en les séparant par des point-virgules.

```
$ who ; date ; ls
```

6.7. Enchaînement complexe

Vous pouvez enchaîner des commandes en indiquant des conditions d'exécutions. Le symbole && permet d'exécuter la deuxième commande si et seulement si la première à réussie.

Le symbole || permet d'exécuter la deuxième commande si et seulement si la première à échouée.

Si le fichier f1 existe il est renommé en f2 et pwd est exécutée :

```
$ mv f1 f2 && pwd
$
```

6.8. Ordre de priorité

Vous pouvez établir des propriétés grâce aux parenthèses comme pour un calcul.

```
$ (ls | lp) && pwd
$
```

6.9. Les filtres

Quand vous enchaînez des commandes avec le symbole | vous envoyez vos commandes dans des filtres comme **more** ou **pr**.

Le filtre more permet d'avoir des pages écran.

Le filtre pr permet d'avoir une pagination

Impression du contenu du répertoire avec une pagination :

```
$ ls | pr | lp
$
```

7. LES IMPRESSIONS

7.1. Impression de fichier

Lorsque vous imprimez un fichier celui ne va directement à l'imprimante mais passe par une étape intermédiaire qui est le spooler. Le spooler est un répertoire : `usr/spool/lp/request/nomimprimante`. Le service d'impression est le `lpshed`.

La commande **lp** envoie une requête à une imprimante. Elle affiche ensuite le numéro de requête sous la forme `imprimante-nn`

Syntaxe `lp paramètres fichier1`

Paramètres

- c Effectue des copies des fichiers à imprimer
- d<dest> Précise la destination de l'impression (dest est le nom de l'imprimante)
- n<nombre> Imprime un certain nombre de copies.
- o<option> Permet de préciser certaines options d'impression
 noheader pas d'impression d'entête.
 nofilebreak pas de saut de page entre deux fichiers.
- t <titre> Imprime un titre sur la page poster.

Impression du fichier clients :

```
$ lp clients
$
```

Impression sur l'imprimante `imp1` de tous les fichiers du répertoire courant:

```
$ lp -d imp1 *
$
```

7.2. Annulation d'impression

La commande **cancel** annule une requête d'imprimante. Si le numéro de requête n'est pas fourni mais seulement l'imprimante, la commande `cancel` stoppe la requête en cours d'impression.

Syntaxe `cancel <identification requête>`
 `cancel <imprimante>`

Annulation de la requête n°10 sur l'imprimante 1:

```
$ cancel imp1-10
$
```

Annulation de la requête en cours d'impression sur l'imprimante 1:

```
$ cancel imp1
```

7.3. Activation/Déactivation d'une file d'attente

Vous pouvez facilement suspendre l'impression par la commande **disable** et la reprendre par la commande **enable**.

Pour suspendre la file laser :

```
$ disable laser
$
```

Pour reprendre :

```
$ enable laser
$
```

7.4. Obtention de l'état de l'imprimante

La commande **lpstat** affiche le statut de l'imprimante. Avant de lancer une quelconque impression, il convient de connaître l'état du système. Quel sont les imprimantes existantes, sont-elles disponibles? En cours d'impression il est également important de connaître les requêtes en attente, afin de pouvoir éventuellement les supprimer.

Syntaxe lpstat

Paramètres

- a Indique si les imprimantes sont prêtes.
- p Indique l'état des imprimantes.
- d Indique l'imprimante par défaut du système.
- t Affichage complet.
- u Indique toutes les requêtes d'un ou de tous les utilisateurs.
- v Indique les rattachements physiques d'une ou de toutes les imprimantes.

Les imprimantes du système (imp1 et imp2) acceptent-elles des requêtes ?

8. MESSAGERIE

8.1. Envoyer du courrier

La commande **mail** vous permet d'expédier ou de lire votre courrier.

Attention vous quittez le shell pour arriver dans le shell de la messagerie avec ses propres commandes Dans ce cas le prompt change.

Syntaxe mail nomutilisateur

Si la commande mail est saisie sans nom, lecture du courrier. Sinon envoi d'un courrier à l'utilisateur nommé.

Lors de la lecture du courrier, celui-ci est affiché message après message.

Après l'affichage d'un message, il est possible d'employer les commandes suivantes

Entrée	Passage au message suivant.
d	Effacement du message et passage au suivant.
P	Ré-affichage du message.
-	Retour au message précédent.
s	Sauvegarde le message dans un fichier (par défaut mbox).
w	Identique à s mais sans le titre du message.
Ctrl-d ou q	Abandon de lecture sans effacement de message.
! + commande	Exécution d'une commande shell.
?	Affichage d'un résumé des commandes de mail

8.2. Conversation en temps réel

La commande **write** permet l'envoi de message à un utilisateur. Il faut que l'utilisateur en question ait autorisé la réception de message et qu'il soit connecté.

Syntaxe write nomutilisateur

Il faut terminer la saisie du message par ctrl-d.

Envoi d'un message à Marc:

8.3. *Autorisation de réception de message*

La commande **mesg** permet d'autoriser ou non l'arrivée de messages sur la console. Sans paramètres, cette commande affiche l'état courant.

Syntaxe mesg yn

Paramètres

y Autorise l'arrivée de messages.

n N'autorise pas l'arrivée de messages.

Affichage de l'état courant des messages puis interdiction de réception de messages:

```
$ mesg
is y
$ mesg n
$ mesg
is n
$
```

9. LES PROCESSUS

9.1. Introduction

Un processus est une commande qui est en cours d'exécution dans la RAM de la machine.

Attention vous ne pouvez manipuler que les processus dont vous êtes le propriétaire.

Pour avoir la liste de vos processus, il suffit de taper la commande **ps**. L'affichage se présente sous forme de colonne. La description de ces colonnes est la suivante :

<i>N°COL</i>	<i>NOM</i>	<i>DESIGNATION</i>
1	UID	Nom du propriétaire.
2	PID	N° du processus qui l'identifie.
3	PPID	N° du processus qui est le père du processus en cours.
4	C	Basculer des tâches
5	STIME	Heure de départ
6	TTY	Nom du terminal de lancement
7	TIME	Temps cumulé d'exécution
8	COMMAND	Nom de la commande lancée

Cette commande peut avoir les paramètres suivants :

- e Affichage de tous les processus. Colonne 2, 6, 7 et 8.
- f Affichage des processus de la session. Toutes les colonnes.

Affichage des processus en cours :

9.2. Lancement d'un processus en tâche de fond

Tout processus peut être lancé en arrière-plan, afin de rendre immédiatement la main à l'utilisateur.

Pour lancer un processus en tâche de fond, ajoutez **&** en fin de ligne. Le système affiche alors le numéro de processus créé et rend la main instantanément. Vous pouvez ajouter une redirection de sortie vers le fichier factice `/dev/null` afin d'éviter les messages parasites sur l'écran.

9.3. Destruction d'un processus

La commande kill permet de détruire un processus. Attention on ne peut détruire que les processus dont on est propriétaire.

Syntaxe kill -9 N°PID du processus

Destruction du processus 515 :

```
$ kill -9 515
$
```

9.4. Programmation d'un processus

Vous pouvez programmer le lancement d'une commande à une heure et un jour précis. C'est la commande **at**. Utilisée sans argument, vous obtenez la liste des commandes programmés en cours.

Syntaxe at time date
 at -r n° travail
 at -l n° travail

Paramètres

time heure sous la forme HHSS ou HH:MMam ou HH:MMpm
date date sous la forme MMM JJ
-r n° travail suppression d'un travail donné.
-l n°travail affichage des travaux en attente.

Attention il faut que les services de programmation soit en route sur la machine.

Exécution du script monfichier à 15 H 41, puis exécution de autrefichier dans une heure:

```
$ at 1541 < monfichier
$
$ at now + 1 hours < autrefichier
```

10. L'EDITEUR VI

10.1. Introduction

L'éditeur de texte **vi** est l'éditeur standard pleine page d'UNIX. Sa convivialité est pour le moins restreinte, mais il reste néanmoins un outil puissant. Orienté page, il doit impérativement connaître votre type de terminal en accédant à la variable d'environnement \$TERM.

Il fonctionne dans trois modes : le mode insertion (saisie de texte), le mode commande et le mode de commande **ed**.

Pour activer le mode insertion, tapez une des commandes d'insertion citées ci-après. Le mode insertion sera conservé jusqu'à la frappe de **Echap** ou jusqu'à une fin d'insertion signalée par un bip sonore (ou un flash écran).

Si vous doutez du mode actif, appuyez sur **Echap** pour activer le mode commande.

Le mode de commande **ed** est obtenu par la frappe des deux points en mode commande.

10.2. Lancement et fin de vi

Pour lancer l'éditeur **vi** il suffit de taper :

```
vi nomfichier
```

Pour quitter l'éditeur **vi** on tape **ZZ**.

10.3. Commande d'insertion caractères/lignes

i	Passage en mode insertion devant le curseur.
A	Passage en mode insertion en fin de ligne.
a	Passage en mode insertion après la position du curseur.
O	Passage en mode insertion d'une ligne entière avant la ligne courante.
o	Passage en mode insertion d'une ligne entière après la ligne courante.
Esc	Fin d'insertion, passage en mode commande.

10.4. Touches de déplacement

Ctrl n	Bas
Ctrl p	Haut. Si vous avez les touches des flèches, utilisées.
Ctrl u	Scrolling écran vers le haut.
Ctrl d	Scrolling écran vers le bas.
Ctrl f	Page suivante.
Ctrl b	Page précédente.
w	Mot suivant.
b	Mot précédent.
E	Fin de mot.
{	Début de paragraphe.
}	Fin de paragraphe.
(Début de phrase.
)	Fin de phrase.
M	Milieu d'écran.
H	Haut d'écran.
L	Bas d'écran.
0	Début de ligne.
\$	Fin de ligne.
<n>G	Va à la ligne <n>.
:<n>	Va à la ligne <n>.

10.5. Commandes de suppression de texte

x	Destruction d'un caractère sous le curseur.
X	Destruction d'un caractère à gauche du curseur.
Dd	Destruction de la ligne courante
<n>dd	Destruction de <n> lignes.
dw	Destruction du mot courant.
<n>dw	Destruction de <n> mots.
D	Destruction de la fin de la ligne.

10.6. Insertion des tampons

Les commandes de suppression ne détruisent pas irrémédiablement le texte visé. Il est stocké dans un tampon (buffer), le tampon standard. Celui-ci peut être inséré tant qu'une nouvelle commande de suppression n'a pas été lancée. Il existe également différents tampons nommés de "a" à "z" qui peuvent être utilisés pour stocker provisoirement du texte. Il est possible de copier une portion de texte vers un tampon quelconque sans pour autant le supprimer.

p	Insertion du texte contenu dans le tampon standard après le curseur.
P	Insertion du texte contenu dans le tampon standard avant le curseur.
yw	Copie du mot courant dans le tampon.
Y	Copie de la ligne courante dans le tampon.
nY	Copie de n lignes dans le tampon.
"anY	Copie de n lignes dans le tampon a.
"ap	Insertion du tampon a après le curseur.

10.7. Commandes d'annulation/répétition

u	undo (annulation).
.	répétition d'action

10.8. Recherche et remplacement d'occurrences

/<ch1>	Recherche la chaîne <ch1>
N et n	Continue la recherche dans l'une ou l'autre direction.
:s/<ch1>/<ch2>	Remplace la chaîne <ch1> par la chaîne <ch2>.

10.9. Commandes d'entrées/sorties

:q	Quitte vi si aucune modification n'a été effectuée.
:q!	Quitte vi même si des modifications ont eu lieu.
:w	Ecrit sur disque.
:r <nom>	Lit le fichier <nom>.

10.10. Commandes diverses

!: cmd	Exécute la commande cmd. Retour à vi par Ctrl-d.
:set nu	Numérote les lignes
:set nonu	Enlève les n° des lignes

11. LE BOURNE SHELL

11.1. Lancement

Vous pouvez lancer le bourne shell par la commande sh. A ce lancement il y a exécution du fichier .profile dans votre répertoire privé à la connexion seulement et non lancement du shell.

11.2. Caractères spéciaux

11.2.1. Génération de noms de fichiers

Lors d'actions sur les fichiers comme les copies, impressions, suppression ou autre, il est possible d'indiquer au Shell que l'on souhaite intervenir sur plusieurs fichiers simultanément.

Le caractère * remplace une suite quelconque de caractères. Il agit à la façon d'un joker. Il peut se trouver placé au début ou à la fin d'un nom de fichier (contrairement à Msdos).

Liste de tous les fichiers commençant par texte:

```
$ ls texte*
texte1
texte2
texte3
textejean
$
```

Le caractère ? remplace un seul caractère quelconque.

Recherche des fichiers toto titi tutu...:

```
$ ls t??
toto
titi
tutu
$
```

Les caractères [] indiquent une énumération parmi laquelle pourra être choisi un caractère pour la substitution.

Pour rechercher tous les fichiers dont le nom commence par texte et se finit par un chiffre compris entre 1 et 4:

```
$ ls texte[1-4]
texte1
texte2
texte3
texte4
$
```

11.3. Quotation des chaînes et des commandes

Les simples quotes doivent être utilisées si le texte saisi contient des méta-caractères ne devant pas être évalués.

```
$ nom=bruno
$ echo 'Bonjour'
Bonjour $nom
$
```

Les doubles quotes agissent comme les simples quotes mais autorisent l'expansion des trois méta-caractères suivants \$ (variables shell) ' (backquotes) et \ (antislash).

Affichage d'une chaîne comportant une partie fixe et une partie variable:

```
$ nom = bruno
$ echo "Bonjour $nom"
Bonjour bruno
$
```

Les backquotes ` permettent l'évaluation d'une commande et fournissent le résultat de celle-ci.

Stockage du résultat de la commande logname dans la variable monnom:

```
$ monnom = `logname`
$ echo $monnom
marc
$
```

L'antislash \ annule l'interprétation du caractère qui suit.

Affichage de l'astérisque en tant que caractère, celui-ci ne sera pas interprété comme signifiant tous les fichiers:

```
$ echo voici un astérisque : \*
Voici un astérisque : *
$
```

11.4. Programmation en Boume shell

Pour créer un programme en Boume shell, il faut créer un texte contenant la liste des commandes que doit exécuter le programme. Ce texte sera ensuite rendu exécutable par l'instruction chmod et en plaçant un x.

Attention pensez au chemin de recherche défini par la variable PATH

Pour placez un commentaire vous devez mettre le caractère #

11.4.1. Les Variables

Les variables Bourne shell sont de type caractère seulement. Il n'est donc pas possible d'effectuer des calculs portant sur des variables.

Un nom de variable doit débuter par une lettre suivie d'un ou plusieurs caractères alphanumériques. Il peut être en majuscules ou en minuscules, mais fait alors référence à des objets différents.

Pour affecter une variable, on doit utiliser la forme suivante sans aucun espace de chaque côté du signe d'affectation = : variable = texte

```
$essai = Bonjour
$ echo $essai
Bonjour
$
```

11.4.2. Les variables d'environnement

Certaines variables font partie de l'environnement du shell et sont en principe écrites en majuscules.

VARIABLE	DESCRIPTION
\$CDPATH	Contient la liste des répertoires à scruter pour la commande cd.
\$HOME	Contient le chemin d'accès au répertoire de l'utilisateur.
\$MAIL	Contient le nom du fichier courrier de l'utilisateur.
\$PATH	Contient la liste des répertoires à scruter pour la recherche d'une commande exécutable. Les répertoires sont séparés par le caractère deux-points (:).
\$PS1	Contient la valeur du prompt
\$PS2	Contient la valeur du prompt secondaire quand une commande n'est pas achevée d'être frappée.
\$TERM	Contient le nom du terminal

12. LE C SHELL

12.1. Lancement du C shell

Lancement du C shell en recouvrement ou non :

```
Syntaxe    cshell
           exec cshell
           csh
```

Le C shell est un interpréteur de commandes dont la syntaxe se rapproche plus de celle du langage C que le Bourne shell. Son chemin d'accès réel est /bin/csh. Très utilisé, il offre davantage de confort d'utilisation que le Bourne shell.

A chaque invocation du C shell, le fichier .cshrc de l'utilisateur est exécuté, alors que le fichier .login l'est seulement à la connexion et le fichier .logout seulement à la déconnexion.

12.2. Mécanismes d'historique

Il est possible en C shell de manipuler les dernières commandes utilisées. Celles-ci sont repérées par un numéro d'ordre croissant au fur et à mesure de la frappe des commandes.

Voici comment faire :

history	Affichage des dernières commandes utilisées.
!!	exécution de la dernière commande frappée.
!numéro	Exécution d'une commande selon son numéro.

12.3. Création et suppression d'alias

12.3.1. Création d'un alias

Les alias permettent de renommer une commande.

Le C shell autorise la création d'alias aux commandes. Le mécanisme d'historique est disponible pour la création d'un alias.

```
Syntaxe    alias <définition de l'alias>
```

Création des alias dir et lm

```
% alias dir ls
% alias lm 'ls -l \!* | more'
```

Attention les alias n'existent que dans le C shell en cours sauf sous Linux mais en Bourne shell il faut taper *alias dir=ls*.

La commande alias sans paramètre vous donne la liste des alias en cours.

12.3.2. Suppression d'un alias

Pour supprimer un alias existant il suffit de taper la commande suivante :

```
unalias nomalias
```

Suppression de l'alias dir :

```
% unalias dir  
%
```

12.4. Programmation en C shell

Le C shell est un véritable langage de programmation, autorisant les manipulations de variables, l'accès à toutes les commandes d'Unix et même les sessions interactives avec interrogation et saisie de réponses, mais sort du cadre de ce support.

Comme en bourne shell, un programme C shell est un texte rendu exécutable par la commande `chmod` et en plaçant un `x`.

Toute ligne de programme commençant par `#` (dièse) ne sera pas exécutée.

La première ligne d'un programme doit impérativement comporter un dièse pour indiquer que le programme doit être exécuté en C shell et non en Bourne shell.

12.5. Les variables

Les noms de variables peuvent comporter jusqu'à 20 caractères, incluant chiffres, lettres ou soulignés. Il existe trois sortes de variables : les variables ordinaires, créées par l'utilisateur, les variables prédéfinies, les variables d'environnement.

12.5.1. Les variables ordinaires

Elle peuvent être affectées à l'aide de la commande **set** pour les variables caractères ou à l'aide de **@** pour les variables numériques. Elles sont détruites à l'aide de la commande **unset**.

La commande **set** sans paramètres affiche les variables actuellement définies.

```
Syntaxe    set  
           set nomvariable = "chaine"  
           unset nomvariable  
           @ nomvariable = valeur
```

Affectation de variables alphanumériques et variables numériques. Calcul sur des valeurs numériques:

12.5.2. Les variables prédéfinies

En général, ces variables définissent certains paramètres du shell.

NOM	DESCRIPTION
\$argv	Contient la liste des paramètres reçus par un programme C shell.
\$cdpath	Indique le chemin de recherche alternatif pour la commande cd.
\$child	Contient le numéro du dernier processus lancé en arrière-plan. Cette variable est éliminée par unset lorsque la tâche se termine.
\$echo	Affiche la commande exécutée avant son exécution
\$history	Indique le nombre de commandes mémorisées (set history=10).
\$home	répertoire utilisateur. Le méta-caractère ~ fait référence à cette variable.
\$ignoreeof	Empêche la terminaison du shell par Ctrl-d
\$noclobber	Empêche une redirection vers un fichier.
\$path	Indique les chemins de recherche d'exécution.

Attention il ne faut pas confondre avec les variables d'environnements que vous allez voir dans le paragraphe suivant.

12.5.3. Les variables d'environnement

Ces variables créées par le shell sont utilisées par les programmes lancés pour déterminer leur comportement.

Elles sont en majuscules et peuvent être affichées par la commande **env**.

Elles peuvent être positionnées par la commande **setenv**.

NOM	DESCRIPTION
\$TERM	Cette variable contient le nom du terminal utilisé, qui doit être l'un de ceux prévus dans terminfo.
\$PATH	Cette variable contient le chemin d'accès aux fichiers exécutables.
\$CDPATH	Liste de répertoires alternatifs pour la commande cd.
\$LOGNAME	Nom de connexion.
\$SHELL	Shell utilisé (sh ou csh ou ksh).
\$HOME	Répertoire utilisateur.
\$MAIL	Chemin d'accès au fichier de messagerie.