Université Aix-Marseille I — **Maîtrise de Biochimie**

Université Aix-Marseille II — **Maîtrise de Biologie Cellulaire**

Abécédaire UNIX

Bernard Bellon 1994

Introduction

Le système UNIX a été créé vers les années 1970 dans les laboratoires de recherche de la firme A.T. T. Ce système a subi depuis de nombreux changements, en particulier sa réécriture en langage C. Deux versions constituent la référence de la quasi totalité des systèmes existants à ce jour, la version des laboratoires A.T & T. et la version de l'Université de Berkeley dont la différence essentielle se situe au niveau de la gestion de la mémoire et au niveau de la communication entre processus. L'une des évolutions majeures d'UNIX depuis ses origines a été l'intégration du système X-window dans l'environnement.

Les caractéristiques d'UNIX sont un système d'exploitation multi-tâches et multi-utilisateurs et indépendant de toute architecture de machine et de tout constructeur. Sa conception est modulaire et basée sur trois couches bien définies :

- le noyau qui comprend toutes les fonctions nécessaires d'un système d'exploitation (allocation du processeur, gestion mémoire, gestion des ressources, entrées/sorties ..)
- l'interface utilisateur, appelée *shell* qui remplit un double rôle, celui d'interpréteur de commandes et celui d'un langage de programmation
- l'ensemble des outils disponibles tels que les langages de programmation et leurs bibliothèques, les outils spécifiques de développement dans un environnement particulier (Xwindow).

Les deux facettes du système UNIX les plus apparentes à l'utilisateur sont le système de fichiers et l'interpréteur de commandes (shell). La suite de ce polycop n'est pas un résumé des diverses commandes UNIX, mais simplement le nécessaire qui doit vous permettre d'utiliser le système et l'ensemble des logiciels de biologie implantés sur ces stations de travail. Pour les commandes décrites, seules les options les plus utiles vous seront présentées. Les paragraphes bordés d'un double trait sont d'une compréhension indispensable.

1 - Le système de gestion des fichiers

Le système n'impose aucune structure particulière aux fichiers sauf pour certains le concernant directement. Il ne fait aucune supposition quant à leur contenu et leur utilisation, cela ne regarde que les programmes qui construisent ou qui exploitent les fichiers. Chaque fichier est décrit par un enregistrement appelé *i-node* qui précise son type, sa taille, la date de création et de sa dernière modification, le numéro du propriétaire et du groupe, les permissions, les liens .... Les i-nodes sont rangés dans une table et chacun est identifié par un numéro.

UNIX distingue trois sortes de fichiers :

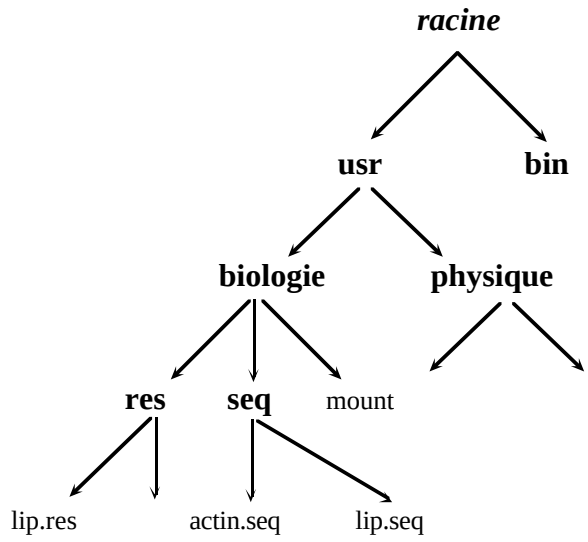
- le type ***répertoire***, ou *catalogue* (directory) est un fichier dont le contenu est une liste de couples (nom, numéro d'un i-node) dont chacun est un renvoi à un fichier. Les répertoires définissent un structure d'arbres sur l'ensemble des noms de fichiers. Tout fichier est donc accessible par au moins un nom appartenant à un répertoire, sauf un fichier particulier de type répertoire qui est la racine et qui n'a pas de nom.

- le type ***fichier spécial*** qui correspond aux unités matérielles d'entrée/sortie. Du point de vue du programmeur, les procédures de lecture ou écriture sont les mêmes que celles qui permettent l'échange d'information avec un périphérique quelconque.

- le type ***fichier ordinaire*** qui correspond à tous les autres fichiers. Pour l'utilisateur, deux distinctions que le système ignore : les fichiers de texte qui sont éditables, imprimables, affichables et les fichiers binaires qui ne le sont pas.

1.1 - Les noms de fichiers

De nombreuses commandes du shell ainsi que les programmes font référence à des noms de fichier pour trouver l'information désirée. Il est important de savoir écrire correctement un nom de fichier et en particulier dans les deux écritures référence absolue et référence relative. La manière de nommer les noms de fichier utilise la structure arborescente des répertoires avec comme caractère séparateur le caractère "slash" /.



Exemple d'arborescence :

les noms de répertoire sont écrits en gras

Nous allons expliciter les divers types de référence en nous appuyant sur cet exemple. Tout

nom de fichier figure dans un répertoire qui, s'il n'est pas la racine, figure à son tour dans un autre répertoire.

1.1.1 - Références absolues

La référence absolue d'un nom de fichier s'écrit à l'aide du chemin qui mène au fichier considéré en partant de la racine : le chemin absolu s'écrira (racine)/usr/biologie/seq/ et le fichier "lip.seq" sera désigné par (racine)/usr/biologie/seq/lip.seq. Le répertoire racine n'ayant pas de nom nous l'écrirons **/usr/biologie/seq/lip.seq**. Le caractère "/" tout seul est une référence absolue à la racine du système de fichiers.

1.1.2 - Références relatives

Pour alléger les références aux fichiers, un répertoire joue un rôle particulier, c'est le répertoire de travail. Un chemin qui ne commence pas par le caractère "/" est un chemin relatif : son premier élément est recherché dans le répertoire de travail. Supposons que le répertoire de travail soit biologie; le chemin relatif menant au fichier lip.seq est seq, et le fichier "lip.seq" sera désigné par **seq/lip.seq**

1.1.3 - Auto-références

Chaque répertoire contient deux noms spéciaux avec un rôle prédéfini. "." renvoie au répertoire lui-même et ".." renvoie au répertoire ancêtre ou père. Les chemins peuvent être exprimés à l'aide de ces deux références particulières.
Le fichier "lip.seq" peut être désigné par **./seq/lip.seq** si le répertoire de travail est biologie.
Le fichier "lip.res" peut être désigné par **../res/lip.res** si le répertoire de travail est seq.

1.2 - Les permissions

Tout utilisateur d'un système UNIX possède un numéro d'utilisateur unique et un numéro de groupe commun à plusieurs utilisateurs. Les permissions des fichiers (lecture, écriture, exécution) sont définies relativement à trois domaines : le propriétaire du fichier, le groupe comprenant les utilisateurs du même groupe que le propriétaire, et les autres. Les permissions sont listées dans l'ordre propriétaire, groupe, autres et pour chaque domaine sont écrites la permission de lecture (r), écriture (w) et exécution (x). Lorsque la permission n'est pas accordée un tiret remplace la lettre correspondante. Par exemple "rwxr-x--x" indique un fichier sur lequel le propriétaire a tous les droits, le groupe ceux de lire et d'exécuter, et les autres celui d'exécuter.
Bien évidemment, le propriétaire a seul le pouvoir de modifier les permissions.
Chaque programme ou logiciel attribue des permissions spécifiques lors de la création de fichiers.

2 - Le shell, l'indispensable

Le shell est l'interprète des commandes d'UNIX, c'est un programme qui dialogue avec tout utilisateur. Dès que la procédure de connexion (login) est terminé, il prend le contrôle, lit les commandes que vous tapez, les interprète et lance les actions adéquates. Plusieurs interprètes sont disponibles : le Korn shell, le Bourne shell et le C shell. Sur les stations que vous utiliserez, les trois shells sont disponibles et par défaut lorsque vous établirez une connexion, c'est le processus C shell qui sera activé (défini par l'administrateur du système).

Toute activité sous UNIX s'exécute dans le cadre d'un processus. Tout processus (appelé fils) est créé par un autre processus (appelé père). Par exemple lorsque vous lancerez une commande permettant d'utiliser votre terminal comme un terminal X (fenêtre, souris, menus..), vous lancerez un processus. Lorsque vous lancez une commande classique, vous créez un processus fils et ne revenez au processus père qu'une fois la commande terminée. Une commande peut être lancée en arrière plan (mode détaché) de telle façon que le processus va se dérouler en parallèle vis à vis du processus père. Après le lancement de la commande en arrière plan, vous revenez tout de suite au processus père sans attendre que le processus fils soit terminé. Lancer une commande en arrière plan s'obtient en écrivant la commande suivie d'un espace et d'un et commercial "&".

2.1 - Une session

Les stations de travail dont vous disposez **ne doivent jamais être éteintes**. Ne touchez donc jamais l'interrupteur situé sur la face avant de l'unité. Seul l'écran ou moniteur peut être éteint, dans ce cas le voyant de marche n'est pas éclairé. Pour allumer l'écran, basculez l'interrupteur situé au bas de l'écran.

Si sur un poste donné, aucun utilisateur n'est connecté, alors l'écran s'il est allumé doit afficher une invite à la connexion se terminant par le mot "login :". Si ce n'est la cas appuyez sur la touche chariot-retour (return ou Entrée).

2.1.1 - Début d'une session

Pour utiliser les machines, vous devez disposer d'un nom d'utilisateur (user name) et d'un mot de passe (password) qui vous ont été attribués.

Au prompt de connexion, tapez votre nom d'utilisateur en minuscule puis chariot-retour. Le système vous demandera votre mot de passe (Password). Tapez ce dernier puis chariot-retour, en faisant attention, car vous n'aurez plus d'écho sur l'écran et vous taperez donc en aveugle (secret en principe). Si la connexion réussit, le système affiche quelques informations (message de l'administrateur, date de dernière connexion, courrier électronique (mail)) et le prompt correspondant au shell (% sous C shell). Dans le cas d'un échec le prompt de

connexion est de nouveau affiché, faites une autre tentative.

Si la connexion a réussi, vous êtes sous C shell devant un terminal de type TTY. Il sera pour vous beaucoup plus facile de travailler sous Xwindow. Pour cela vous lancerez la commande **open xinit** qui va créer un shell et mettre le système sous type AIX.

Vous obtiendrez une fenêtre intitulée "aixterm". Cette fenêtre servira de base pour créer les divers processus et en particulier les nouvelles fenêtres dans lesquelles vous travaillerez. N'oubliez pas de créer toutes vos fenêtres comme processus fils de aixterm et en arrière plan. Pour cela vous écrirez au prompt de la fenêtre "aixterm" la commande :

aixterm -T travail -sb & ---> -T doit être suivi du titre de la fenêtre, -sb indique la présence de barres de défilement et & crée le processus en arrière plan

2.1.2 - Fin d'une session

La fin d'une session est lancée par la commande logout sous C shell, ou encore les commandes Ctrl D ou exit. La déconnexion ne peut avoir lieu que sous le contrôle du shell de connexion. Il est impératif d'arrêter tous les processus ou tâches que vous avez en cours avant de demander une déconnexion.

Si vous êtes sous Xwindow (ou AIX), vous arrêterez tous les programmes en cours, n'oubliez pas l'éditeur (cela est très fréquent), vous arrêterez tous les processus attachés à chacune des fenêtres en fermant celles-ci, sauf le shell père correspondant à la fenêtre de titre "aixterm". En appuyant sur le bouton de la souris, lorsque cette dernière est à l'extérieur de la fenêtre, un menu vous sera proposé avec un item "Sortie", choisissez ce dernier, un message va apparaître vous demandant de confirmer la sortie de "Wmg", confirmez et vous voilà dans le shell de connexion avec l'écran TTY. Tapez la commande logout ou exit ou Ctrl D.

Remarque : *si vous essayez de vous déconnecter au niveau de Xwindow, le système refusera en envoyant le message "Ceci n'est pas un shell de connexion". Sous système UNIX IBM, appuyer sur les touches AltGr et Ctrl/Val vous permet de permuter sur les shells existants. Il peut donc vous arriver de passer brutalement du shell Xwindow au shell de connexion, ne criez pas "j'ai perdu mes fenêtres", appuyez sur les deux touches et vous retomberez dans le shell "fenêtres", surtout ne créez pas un nouveau shell aixterm par la commande* ***open xinit***.

2.2 - Principe général des commandes

La totalité des commandes prédéfinies (et la plupart du temps des programmes) suivent les conventions suivantes :

- le nom d'une commande est suivi par un nombre quelconque d'options de la forme
  - ° -caractère --> option qui n'exige aucune valeur
  - ° -caractère valeur --> option qui demande une valeur

- si aucune option ne comporte de valeurs, on doit les regrouper
- l'ordre des noms de fichier dépend de la nature de la commande

2.2.1 - Noms de fichier génériques

De nombreuses commandes du shell ont des noms de fichiers pour argument. Ces noms peuvent comporter des spécifications génériques :

- ***** --> n'importe quelle chaîne de caractère
- **?** --> n'importe quel caractère unique
- **[..]** --> n'importe quel caractère unique parmi ceux de l'ensemble

2.2.2 - Déroutement des entrées/sorties, tubes

L'entrée ou la sortie standard ne sont pas fixées lors de l'écriture de commandes. En l'absence d'indication, l'entrée est le clavier et la sortie est l'écran du poste. On peut cependant dérouter l'entrée ou la sortie standard par les spécifications suivantes :

- < fichier --> l'entrée est le fichier et non plus le clavier
- > fichier --> la sortie est le fichier et non plus l'écran. Le fichier est créé lors de l'exécution. Si un fichier de même nom existe, il est irrémédiablement détruit et remplacé.
- >> fichier --> la sortie est le fichier et non plus l'écran. Si le fichier existe déjà, il n'est pas écrasé et l'écriture se fait en fin de fichier à la suite de l'information préexistante.

Un tube (pipe) réalise aussi un déroutement des entrées/sorties en même temps qu'il enchaîne l'exécution de deux commandes. La demande d'un tube est précisée par l'écriture de deux commandes séparées par le caractère barre verticale "|" que vous obtenez sur les claviers des stations IBM par la combinaison des touches AltGr &. Les deux commandes s'effectuent en parallèle mais de manière synchronisée, chaque commande attendant l'autre en cas de besoin.

Par exemple la commande "ls -al" permet de lister tous les fichiers du répertoire courant dans le format "long". Lorsque le nombre de fichiers dépasse le nombre de lignes de votre fenêtre, vous verrez défiler les lignes et vous aurez une liste incomplète. Pour avoir la liste par page écran, il suffit de réaliser un tube qui relie la commande "ls -al" avec la commande "more" qui liste le contenu d'un fichier page par page : **ls -al | more**

3 - Principales commandes

Dans la description des commandes, la construction [expr] indique que expr est optionnelle et la construction expr...expr indique une suite d'un nombre quelconque de expr (zéro y compris).

3.1 - Commandes générales

echo [-n] *argument...argument*

Affiche les arguments sur la sortie standard. Utile pour afficher le contenu de certaines variables de l'environnement.

```
% echo  Bonjour
 Bonjour
% echo  $HOME      (HOME est une variable d'environnement qui donne le
 /usr/biologie                  répertoire de connexion)
```

Le signe $ indique que nous voulons l'affichage de la valeur de la variable HOME

cat *fichier...fichier*

Ecrit sur la sortie standard les fichiers indiqués ou l'entrée standard si aucun nom de fichier n'est donné. Cette commande est très utile pour concaténer une série de fichiers en un seul fichier.

```
% cat  fichier1  fichier2  fichier3  >  fichiergeneral
  (met dans le fichiergeneral le contenu du fichier1, puis celui du fichier2
et enfin celui du fichier3)
% cat  fichier4  fichier5  >>  fichiergeneral
  (ajoute au fichiergeneral le contenu du fichier4, puis celui du fichier5)
```

hostname

Ecrit sur la sortie standard le nom de la machine utilisée. Les machines de la salle portent les noms "ens**n**" où n varie de 1 à 35. Le nom du serveur principal est "ylumin".
Cette commande vous sera très utile lors de la connexion sur le serveur ou sur une machine permettant l'impression de documents.

more *fichier...fichier*

Affiche le contenu des fichiers indiqués sur la sortie standard par page écran. L'utilisateur peut taper :

- un blanc pour l'affichage de la page suivante
- un retour-chariot (ou Entrée) pour afficher une ligne supplémentaire
- q pour quitter le programme more.

whoami

Affiche le nom de l'utilisateur (nom du "user" lors de la connexion)

who

Affiche les noms des utilisateurs connectés à la station de travail.

3.2 - Commandes de manipulation des fichiers

3.2.1 - Les répertoires

pwd

Affiche sur la sortie standard le répertoire de travail. Très utile lors de manipulations de fichiers (copie, délétion, ..)

cd [*chemin*]

Permet de définir le répertoire de travail. Si le "chemin" est omis, la valeur de la variable HOME est utilisée.

```
% cd  /usr/biologie/seq
% pwd        (vérification du répertoire de travail)
 /usr/biologie/seq
```

mkdir *chemin...chemin*

Création des répertoires indiqués.

```
% pwd        (vérification du répertoire de travail)
 /usr/biologie/seq
% mkdir lesactines  (création d'un répertoire /usr/biologie/seq/lesactines)
% mkdir /usr/biologie/seq/lesactines      (commande identique)
```

Lorsque vous utiliserez les stations de travail, vous aurez un nom d'utilisateur commun avec d'autres étudiants, prenez l'habitude de créer au moins un répertoire personnel dans lequel vous sauvegarderez vos fichiers de travail.

rmdir *chemin...chemin*

Destruction des répertoires indiqués. Celle-ci ne peut avoir lieu que si les répertoires sont vides (si ce n'est le cas, une erreur est signalée)

```
% pwd
 /usr/biologie/seq
% ls
drwxr-x---   2     ber   biol  64    Jun  21     15:32       lesactines
-rwxr-x---   1     ber   biol  680   Jun  21     15:32       lip.seq
% rmdir /usr/biologie/seq/lesactines
% ls
-rwxr-x---   1     ber   biol  680   Jun  21     15:32       lip.seq
```

3.2.1 - Les fichiers

ls [*options*] *nom...nom*

Affiche le contenu des répertoires et les informations sur les fichiers indiqués. Si aucun nom n'est indiqué, c'est le contenu du répertoire de travail (ou courant) qui est affiché.
De nombreuses options existent, les plus utiles sont les suivantes :

- a affiche tous les fichiers y compris ceux commençant par un point
- l format de sortie long : dans l'ordre nous avons le type (d pour répertoire), les permissions, le nombre de liens pour un fichier ou le nombre d'entrées pour un répertoire, le propriétaire, le groupe, la taille en octets, la date de dernière modification et enfin le nom du fichier.
- R affichage récursif de tous les sous-répertoires

```
% cd /usr/biologie
% ls  -al
 -rwxr-x--x  1     ber   biol  820   Jul  28     14:40       mount
drwxr-x---   7     ber   biol  120   Jun  21     15:32       res
drwxr-x---   12    ber   biol  160   May 22      20:17       seq
% cd  /usr/biologie/seq
% ls  -al  *.seq  (affiche tous les fichiers du répertoire courant dont le nom se termine par .seq)
```

cp [*options*] *fichier1 fichier2*
cp [*options*] *fichier...fichier répertoire*

Cette commande permet de copier des fichiers. La première forme permet de copier le fichier1 sur le fichier2 (si fichier2 existe déjà, il est écrasé). Dans la deuxième forme, les fichiers sont copiés dans le répertoire indiqué, chacun conservant son nom.

Les deux options suivantes sont reconnues :

-i	demande de confirmation pour chaque copie
-r	si l'une des références source est un répertoire, la recopie est récursive pour cette référence (copie des sous-répertoire)

```
% cp /usr/biologie/mount  /usr/biologie/seq/toto
 (le fichier mount du répertoire /usr/biologie est copié en fichier toto du
répertoire /usr/biologie/seq)
% pwd
 /usr/biologie
% cp mount  /usr/biologie/seq/toto  (commande identique à la précédente)
% cp mount   seq/toto               (commande identique à la précédente)
% pwd
 /usr/biologie/seq
% cp *.seq   /usr/biologie/res  (copie tous les fichiers du répertoire
/usr/biologie/seq dont le nom se termine par .seq dans le répertoire
/usr/biologie/res)
```

mv *fichier1 fichier2*
mv *fichier...fichier répertoire*

Cette commande permet de changer le nom de fichiers. La première forme permet de changer le fichier1 en fichier2 (si fichier2 existe déjà, il est écrasé). Dans la deuxième forme, les fichiers sont déplacés du répertoire d'origine dans le répertoire indiqué, chacun conservant son nom. Attention les fichiers origine n'existent plus. Il ne s'agit pas d'une recopie du contenu mais simplement d'un changement des i-nodes dans l'arborescence du système de fichiers (on ne peut pas utiliser cette commande pour déplacer un fichier d'un volume dans un autre).

rm [*options*] *fichier...fichier*

Supprime les fichiers passés en argument. Cette commande n'agit pas sur les répertoires. Deux options sont très utiles :

-i	demande confirmation avant chaque suppression
-r	traite récursivement tous les sous-répertoires

```
% rm -i -r  *.seq   (supprime tous les fichiers dont le nom se termine par
.seq avec une demande de confirmation et ceci de manière récursive dans
tous les sous-répertoire du répertoire de travail)
```

Attention : l'option de récursivité avec des noms génériques est très utile, mais aussi très dangereuse. Ne taper jamais la commande "rm -r *" sans savoir votre répertoire de travail.

Lorsque vous vous connectez, vous êtes dans un répertoire, la plupart du temps celui-ci contient des fichiers de configuration et de définition de certaines variables. Leur noms

commencent tous par un point ".login", ".cshrc", une commande du style "rm *" dans ce répertoire les supprimeraient et à la prochaine tentative de connexion, vous auriez de sérieux problèmes.

find *chemin...chemin critère*

Cette commande parcourt les arborescences de fichiers ayant pour racine les chemins indiqués à la recherche de fichiers satisfaisant le critère indiqué. Celui-ci est exprimé par la conjonction (et) d'expressions telles que :

-name	*nom*	fichier doit posséder ce nom
-print		produit comme effet de bord (critère toujours vrai) l'écriture du nom du fichier. A mettre en dernière place pour une évaluation après que les autres conditions aient été satisfaites

```
% find  /usr/biologie  -name cellulase.seq  -print
/usr/biologie/seq/cellulase.seq
```

chmod *mode fichier..fichier*

Permet de modifier les permissions de fichiers dont vous êtes le propriétaire. Le mode peut être exprimé de manière absolue en écriture octale ou de manière relative. Cette dernière est la plus simple à exploiter car seuls les attributs spécifiés sont changés :

- une combinaison des lettres u (propriétaire), g (groupe), o (autres). L'absence de paramètres est équivalente à ugo (c'est à dire tous)
- le signe + pour ajouter, - pour enlever et = pour donner ce droit uniquement
- une combinaison des lettres r (lecture), w (écriture), x (exécution)

```
%  ls -l
 -rw-r--r--  1     ber   biol  820   Jul  28     14:40       mount
% chmod  ug+x  mount
%  ls -l
 -rwxr-xr--  1     ber   biol  820   Jul  29     11:20       mount
```

Le fichier mount devient un fichier exécutable par le propriétaire et les membres du groupe.

Les permissions sur les fichiers permettent de gérer les commandes de gestion des fichiers. Les diverses commandes de destruction, de déplacement de fichiers et de répertoire n'agissent que si la commande est exécutée par le propriétaire. Les commandes de copie de fichiers n'agissent que si la commande est exécutée par un utilisateur ayant droit de lecture.

Les créations de fichiers (mkdir, création par un éditeur) attribuent les permissions suivantes par défaut : "(d)rw-r--r--" . Si vous écrivez un script à l'aide du langage C shell par exemple et le sauvegardez dans un fichier, il faudra lui attribuer le mode exécutable pour que le script

soit lancé en tapant le nom du fichier. Les fichiers binaires produits par les compilateurs ont par défaut le mode exécutable.

diff [*options*] *fichier1 fichier2*

Recherche les lignes différentes entre les fichier1 et fichier2. Cette différence est exprimée de la façon suivante :

```
        différence
[       <
        copie des lignes concernées du fichier1  ]
[       >
        copie des lignes concernées du fichier2  ]
```

Une option est fort utile pour supprimer les différences dues au blanc (espace ou tabulation)

-b ignore les blancs

grep [*options*] *expression* [*fichier...fichier*]

Recherche et affiche toutes les lignes des fichiers d'entrée qui contiennent une chaîne de caractères qui s'unifie avec *expression* qui doit être une expression régulière (voir plus loin). Les options disponibles sont :

-c donner uniquement le nombre de lignes satisfaisant à la question

-h ne pas mettre le nom de fichier devant chaque ligne trouvée

-i ne pas distinguer lettres majuscules et minuscules

-n chaque ligne sera précédée de son numéro dans le fichier

4 - Gestion des commandes

commande &

Exécution de la commande en arrière-plan. Le shell lance la commande indiquée et sans en attendre la terminaison réaffiche le prompt pour vous permettre d'exécuter d'autres tâches. Nous avons précédemment utilisé cette option lors de l'ouverture de fenêtres. Dans le cas de l'ouverture de plusieurs fenêtres, cette manière d'opérer permet d'avoir le prompt disponible dans chacune. N'abusez pas de cette option car l'affichage du à l'action d'une commande lancée en arrière-plan s'effectue quand même sur la fenêtre de lancement.

commande1 ; commande2

Les commandes sont exécutées l'une à la suite de l'autre comme si elles avaient été écrites sur des lignes successives.

commande1 && commande2

C'est un enchaînement conditionnel où la commande2 n'est exécutée que dans le cas où la commande1 a réussi (pas d'erreur lors de l'exécution de la commande1)

commande1 || commande2

C'est un enchaînement conditionnel où la commande2 n'est exécutée que dans le cas où la commande1 a échoué (erreur lors de l'exécution de la commande1)

On considère souvent qu'un **tube** (pipe) est un enchaînement de commandes. ce n'est pas exact puisque les deux commandes sont exécutées en parallèle et **la sortie de la première sert d'entrée pour la seconde**.

5 - Editeurs de texte

Plusieurs types d'éditeurs sont disponibles sous UNIX :

- les éditeurs "ligne" qui permettent d'éditer des fichiers avec des requêtes adressées au niveau des lignes. L'éditeur ligne standard sous UNIX est l'éditeur "**ed**" qui n'est pas très convivial mais utilisable quel que soit le type de terminal.
- les éditeurs "écran" qui affichent une partie du fichier sur l'écran et permettent l'édition en utilisant les déplacements du curseur. Parmi ces éditeurs on distingue :
  - ° ceux qui sont par défaut en mode commande
  - ° ceux qui sont par défaut en mode insertion

5.1 - L'éditeur vi

L'un des éditeurs les plus répandus est l'éditeur "**vi**", éditeur "écran" en mode commande. Il a besoin de connaître le type de terminal utilisé, cela ne pose en principe aucun problème pour vous car le type de terminal est défini par la variable d'environnement TERM qui a toujours une valeur définie acceptable lorsque vous êtes simple utilisateur.

vi [*options*] *fichier*

Si fichier existe, l'éditeur "vi" ouvre celui-ci. Si fichier n'existe pas "vi" crée un nouveau fichier.

Nous ne décrirons pas les diverses options qui sont des opérations effectuées avant l'affichage. Cet éditeur a trois modes :

- ***mode commande***	mode initial, tout caractère tapé est interprété comme une commande
- ***mode insertion***	saisie de texte au point courant, le passage en mode commande est obtenu à l'aide de la touche Escape
- ***mode dialogue***	permet la saisie d'une ligne de commande affichée au bas de l'écran. Le caractère retour-chariot marque la fin de la commande qui est alors exécutée et "vi" revient en mode commande

5.1.1 - Les commandes du mode commande

La manière d'avoir la certitude d'être en mode commande est d'appuyer deux fois sur la touche Escape (vous devez entendre au moins une fois la cloche)

- ***Déplacement du curseur***

Le déplacement du curseur a lieu en mode commande. Avec les claviers dont vous disposez, vous pouvez utiliser le pavé de quatre flèches et les flèches barrées.

↑ ou Ctrl P ou k	ligne précédente
↓ ou Ctrl N ou j	ligne suivante
→ ou l	caractère suivant
← ou Ctrl H ou h	caractère précédent
+ ou Return	début ligne suivante
-	début ligne précédente
⇞ ou Ctrl F	écran suivant
⇟ ou Ctrl B	écran précédent
H	positionnement en première ligne de l'écran
L	positionnement en derrière ligne de l'écran
G	positionnement sur la dernière ligne du fichier
w	mot suivant
b	mot précédent
e	fin de mot

- ***Passage en mode insertion***

i	insertion avant le curseur
a	insertion après le curseur
o	ouverture d'une ligne après la ligne courante
O	ouverture d'une ligne avant la ligne courante
I	insertion en début de ligne
A	insertion en fin de ligne
ESC	sortie du mode insertion

Pendant la saisie de texte, la touche Backspace (←) efface le caractère saisi.

- ***Destructions et déplacements de texte***

x	destruction du caractère sous le curseur
X	destruction du caractère précédent le curseur
dd	destruction de la ligne courante
D	destruction de la fin de la ligne courante à partir du curseur
J	juxtaposition de la ligne courante et de la suivante
Y	copie la ligne courante dans le tampon
y$	copie la fin de ligne courante dans le tampon
P	insertion du texte copié (ou détruit) avant le curseur

5.1.2 - Les commandes liées au mode dialogue

L'appel au mode dialogue se fait à partir du mode commande en tapant deux points ":" ou "/" ou "?" selon le style de commande à écrire. L'éditeur affiche une ligne au bas de l'écran où les diverses commandes peuvent être tapées :

: q	sortie de vi
: w *fichier*	sauvegarde du texte dans le fichier (sans nom sauvegarde dans le fichier courant
: r *fichier*	insère à la position courante le contenu du fichier
: q!	sortie de vi sans modification du fichier
/chaîne	recherche vers l'avant de chaîne dans le fichier
?chaîne	recherche vers l'arrière de chaîne dans le fichier
:s...	substitution (utilise les expressions régulières)
:g...	commande globale
:set nu	numérotation des lignes (pour la supprimer : nonu)

D'autres commandes sont disponibles en particulier les commandes globales et celles de substitutions que nous décrirons plus loin car elles utilisent les expressions régulières dont nous parlerons à propos des filtres.

La ligne de commande éditée peut être corrigée à l'aide de la touche Backspace (←).

6 - Les filtres

Un filtre sous UNIX est un programme qui réalise un traitement sur un flot de données en entrée pour fournir un flot de données traitées en sortie. En dehors des filtres simples tels que grep, sort.., il existe des filtres dits "programmables", pour lesquels les actions à réaliser sont décrites dans un pseudo langage. Les deux principaux filtres de cette catégorie sont **sed** (éditeur de flots) et **awk** (traitement de chaînes de caractère).

6.1 - Le filtre sed

Le filtre "sed" est un outil dérivé de l'éditeur "ed" qui s'applique à des flots. Le fonctionnement de sed est le suivant :

- lecture une à une des lignes du fichier d'entrée
- exécution de la commande d'édition
- écriture du résultat sur la sortie standard

sed [-n] [-e *cmde*] [-f *f_cmde*] [*fichier...fichier*]

La commande sed lit les *fichiers* (ou l'entrée standard si aucun nom), leur applique les commandes passées soit dans l'expression *cmde* soit enregistrée dans le fichier *f_cmde* et affiche le résultat sur l'entrée standard. En aucun cas les fichiers d'entrée ne sont modifiés par sed.

L'option -n supprime l'écriture automatique en sortie des informations lues (dans ce cas pour avoir une écriture, il faut le spécifier).

Pour une commande simple, l'indicateur -e peut être omis et celle-ci est le plus souvent écrite entre apostrophe (quote).

La forme générale d'une commande est la suivante : [*sélection*] **fonction** [*arguments*]

Elle décrit la sélection sur laquelle la fonction spécifique agira.

La sélection peut être définie par :

- adr1 [, *adr2*] où adr est un numéro de ligne
- soit par une expression régulière
- si sélection n'est pas précisée, la fonction s'applique à tout le fichier

Certaines commandes comme les fonctions d'insertion ou de délétion ont obligatoirement un terme "sélection", avec pour insertion une sélection définie par un seul numéro de ligne.

```
% sed  's/ala/val/g'  actin.seq >  actinmodif.seq (création d'un fichier
actinmodif.seq où tous les ala ont été remplacées par val)

% sed  -f f_cmde actin.seq >  actinmodif.seq (même résultat si le fichier
f_cmdecontient la ligne s/ala/val/g)
```

Voici quelques fonctions de sed :

/*ch*		recherche la première occurrence de la chaîne *ch*
a*texte*		ajout de *texte* après la sélection
d		suppression à partir de la sélection jusqu'à fin de ligne
D		suppression à partir de la sélection y compris fin de ligne
r	*fich*	lecture du fichier *fich*
w	*fich*	écriture dans le fichier *fich*
s/*a*/*b*/		remplacement de la chaîne de caractères *a* par la chaîne *b*
g		à la fin d'une expression régulière, indique un traitement global
q		fin de sed

La manipulation de chaînes de caractères serait laborieuse dans les commandes de suppression, insertion ou remplacement sans la définition d'expression régulière qui permettent de caractériser des chaînes de caractères avec beaucoup de souplesse.

6.2 - Les expressions régulières

Une expression régulière est un moyen formel pour spécifier une famille de chaînes de caractères (ou motif). Avant de définir complètement celle-ci, il faut définir un certain nombre de caractères susceptibles d'avoir une interprétation particulière.

6.2.1 - Les caractères spéciaux

Les caractères suivants ont une signification particulière sauf lorsqu'ils figurent dans la définition d'un ensemble :

[	indicateur de début de définition d'un ensemble de caractères
.	n'importe quel caractère (unique) sauf fin de ligne
*	indicateur d'un nombre quelconque de répétitions

Les caractères suivants ont une interprétation spéciale dans des contextes particuliers :

]	indicateur de terminaison de la définition d'un ensemble
-	spécifie un intervalle de caractères, n'est plus spécial si en début ou fin de la définition d'un ensemble
^	début de ligne ou complément d'un ensemble s'il suit [
$	fin de ligne lorsqu'il est le dernier caractère d'une expression
\	enlève l'interprétation particulière d'un caractère spécial

6.2.2 - Les expressions régulières atomiques

Il s'agit d'expressions régulières représentant des ensembles de motifs constitués d'un seul caractère.

c	un caractère non spécial quelconque
\s	un caractère spécial précédé de \ qui lui enlève la spécification
[....]	une caractère quelconque appartenant à l'ensemble entre crochets
[^....]	un caractère quelconque n'appartenant pas à l'ensemble entre crochets

6.2.3 - Les expressions régulières

Elles sont de l'une des formes suivantes :

- une expression régulière atomique
- une expression régulière atomique suivie du caractère * définissant tous les mots pouvant être construits sur l'alphabet défini par l'expression
- la concaténation de deux expressions régulières
- une expression comprise entre \(et \) est équivalente à l'expression régulière elle-même mais définit un moyen de repérage (permet d'emboiter les expressions)
- \entier désigne dans une expression le motif correspondant au nième sous-motif d'un motif (sous motif exprimé à l'aide de \(et \))
- une expression régulière atomique suivie de

c\{n\}	exactement n répétitions du caractère c
c\{n,\}	au moins n répétitions du caractère c
c\{n,p\}	entre n et p (bornes incluses) répétitions du caractère c

  Les nombres entiers pris en compte sont $0 < n < 256$

- toute expression dont le premier caractère est ^ indique que le motif doit être en début de ligne
- toute expression dont le dernier caractère est $ indique que le motif doit être en fin de ligne

6.2.4 - Exemples

[a-z]	désigne une lettre minuscule
^[0-9]*	désigne une ligne qui commence par un nombre quelconque de chiffres
^[0-9]\{3\}.[A-Z]	désigne une ligne qui commence par 3 chiffres suivis d'un caractère quelconque lui-même suivi d'une lettre majuscule
[^0-9]	désigne un caractère autre qu'un chiffre
[-^[]	désigne l'un des caractères suivants -, ^, [
^[Bb][a-z]\{4\}	désigne une ligne qui commence par B ou b suivi de 4 lettres minuscules
^\(.*\)\1\1$	désigne des lignes formées de mots répétés deux fois

6.3 - Exemples d'utilisation des filtres sed et grep

Soit un fichier (fich.seq) de séquence protéique dans un format avec numérotation et avec un espace entre chaque aminoacide. J'ai besoin de cette séquence pour lui appliquer un

traitement mais dans un format sans numérotation et sans espace. Je vais donc la transformer à l'aide du filtre (sed). Je peux écrire les commandes suivantes :

```
% sed  's/[0-9]//g'   fich.seq   >  fich1.seq   (substitution des chiffres
                                                 par aucun caractère)
% sed  's/ //g'  fich1.seq  >  snsbfich.seq
ou encore
% sed  's/[0-9]//g'   fich.seq   |  sed  's/ //g'   >  snsbfich.seq
```

Dans cette dernière ligne, une commande de filtre est appliquée au fichier "fich.seq", le flot sortant (traité par cette commande) est à nouveau traité par la deuxième commande de filtre et le flot sortant est redirigé dans la sortie "snsbfich.seq" (fichier).

Le format de la séquence contient des lignes vides et je veux les éliminer :

```
% sed  '/^$/D'   fich.seq   >  snslvfich.seq    (Sélection des lignes
                    contenant aucun caractère puis délétion de la ligne )
```

La séquence est trop longue pour être traitée par un programme et je dois réaliser des traitements partiels en découpant celle-ci en plusieurs morceaux. Je peux utiliser les commandes classiques telles que "**tail**, **head**", mais je peux aussi utiliser le filtre **sed**.

```
% sed '1,100D' fich.seq > l101_fich.seq (garde à partir de la 101ème ligne)
% sed '101,$D' fich.seq > l1_fich.seq     (garde de la ligne 1 à 100)
```

Cette dernière commande peut aussi s'écrire :

```
% sed  '100q'   fich.seq   >  l1_fich.seq
```

Sans afficher ni éditer, je veux savoir si cette séquence contient le motif protéique suivant V ou A, Y, A ou V, G, P". Je vais utiliser le filtre **grep** (cf page 12):

```
% grep  -h  '[AV]Y[AV]GP'  snsbfich.seq   (option -h supprime l'écriture du
                                          nom de fichier à chaque occurrence)
```

Remarque : le système Unix comporte un manuel "en ligne" qui peut être consulté. Vous pouvez obtenir la description et l'utilisation par quelques exemples de la plupart des commandes. Pour cela tapez :

man nom_commande

7 - Communications et réseaux

Il y a quelques années le travail sur ordinateur s'effectuait soit sur un système central par l'intermédiaire d'un terminal soit sur un système personnel. Le premier avait l'avantage d'une capacité de ressource importante mais qui était dégradée par le nombre d'utilisateurs et le second avait l'avantage d'avoir des performances constantes avec toutefois des ressources faibles. Une utilisation différente, basée sur le partage des ressources, a vu le jour ces dernières années grâce au développement des réseaux. Les machines que vous utilisez sont connectées entre elles par un réseau de type local ETHERNET. Celui-ci vous permet de travailler sur une station (utilisation de son processeur, de sa mémoire) en utilisant certaines ressources stockées ailleurs (sur le serveur principal). La gestion de ce réseau est complètement intégrée dans UNIX par le système NFS (Network File System) et NIS (Network Information Service). Un système complet, plus les logiciels de biologie, plus les langages Fortran, Pascal, C occupe au minimum un espace mémoire de masse de 1000 MO. Le poste sur lequel vous travaillez a un disque de capacité de 230 MO, il ne peut être autonome. Le partage en réseau de certaines ressources lui permet de fonctionner comme s'il l'était. De plus la mise en réseau vous permet d'utiliser à partir de votre poste une autre machine. Un autre intérêt du réseau est la circulation de la messagerie électronique. Il existe en France un réseau spécifique recherche et enseignement (Renater) dont l'accès est pour le moment un forfait à l'année. Ce réseau est interconnecté avec les réseaux équivalents des autres pays. Vous pouvez, à condition d'être un utilisateur, travaillé sur une machine située en Allemagne, Angleterre, Etats-Unis ..

Chaque machine appartenant au réseau a un numéro I.P. (Internet Protocol) spécifique et unique déterminé par un codage obéissant à des règles particulières internationales. La machine serveur de la salle est connue de l'extérieur sous le numéro "139.124.11.1" ou encore "ylumin.univ-mrs.fr" (adresse symbolique : celle utilisée couramment car plus facile à mémoriser). Pour le courrier électronique, l'adresse est préfixée par le nom de l'utilisateur, à qui le message est adressé, suivi de @ (at ou arrobas). Par exemple, l'utilisateur actine déclaré sur la machine serveur (ylumin) aura comme adresse "actine@ylumin.univ-mrs.fr".

Nous allons voir uniquement deux commandes dont vous pouvez vous servir, l'une concernant la messagerie, l'autre la connexion sur une autre machine.

telnet *adresse_hote*

Cette commande lance une demande de connexion sur la machine portant l'adresse "adresse_hote". Si l'adresse est correcte le prompt de l'invite de connexion "login" est affiché et vous n'avez plus qu'à effectuer une connexion classique. Une déconnexion s'obtient par la commande quit ou Ctrl D. Si votre adresse est erronée ou tout simplement vous avez tapé la commande telnet (ou tn), le processus est lancé et le prompt suivant est affiché "telnet >".

Dans ce cas ouvrir une connexion s'obtient par la commande "**open** *adrees_hote*" et une déconnexion à ce niveau s'obtient par "**quit**".

```
% hostname
ens23
% tn  ylumin
Trying ...
Connected to ylumin
Escape character is "  "
........                        (procédure de connexion)
% hostname
 ylumin
% Ctrl D
Connexion terminée
% hostname
ens23
```

La configuration de la salle, définie par l'administrateur, est telle que la commande d'impression sur imprimante n'est accessibles que si vous êtes connectés sur une machine particulière, contrairement à une configuration classique.

Lorsque vous aurez besoin d'imprimer, vous vous connecterez sur la machine ens13 ou ens21 et ensuite appliquerez les commandes classiques d'impression. A l'intérieur du réseau local (salle) les adresses telles que ens21 sont suffisantes pour être reconnues.

N'oubliez pas de terminer la connexion avec la machine hôte, une fois le travail terminé.

7.1 - Le courrier électronique

L'utilisation du courrier électronique s'est complètement banalisée depuis l'installation des réseaux. Il existe dans le monde des serveurs gratuits qui vous permettent d'utiliser certains de leurs programmes. Cela ne se fait pas par une connexion par l'intermédiaire de la commande telnet, mais par l'envoi d'un courrier à un utilisateur spécifique dans un format qu'il vous précise. Par exemple, aux Etats-Unis, un serveur vous permet par l'intermédiaire d'un courrier de rechercher des homologies de votre séquence (protéine ou nucléique) dans les banques de séquences biologiques ainsi que d'interroger ces diverses banques.

mail

Cette commande permet d'éditer les messages reçus. Elle affiche le nombre de messages reçus depuis la dernière lecture de la boîte aux lettres puis le prompt particulier du mail "&".

A ce niveau diverses commandes sont disponibles :

& chariot retour	lecture des messages l'un après l'autre
& n	message suivant
& t mess_list	affiche les messages portant les numéros de "mess_list "
& more mess_list	affiche par page les messages
& s mess_list fich	sauvegarde les messages de "mess_list" dans le fichier "fich"
& d mess_list	détruit les messages portant les numéros de "mess_list "
& dp	détruit le message lu et affiche le suivant
& q	quitte en respectant les diverses actions
& x	quitte en laissant la boîte aux lettre dans l'état à l'ouverture

mail *user@adress*

Cette commande permet d'écrire et puis d'envoyer un message à l'utilisateur "user" se trouvant à l'adresse symbolique "adress". Dans le cas d'un utilisateur appartenant au réseau local, le nom d'utilisateur "user" suffit. Après avoir tapé cette commande, vous êtes sous le contrôle de l'éditeur de courrier "mail editor". Vous aurez une invite pour écrire le sujet et le titre puis enfin le message proprement dit. Le texte que vous tapez est le message qui sera envoyé. Pour chaque nouvelle ligne diverses commandes sont disponibles. Elles sont annoncées par le caractère tilde (AltGr =) suivi de la commande:

~r fich	inclut le contenu du fichier "fich"
~f mess_list	inclut les messages reçus portant les numéros de "mess_list
~q	quitte sans envoi de message (message dans dead letter)
~d	inclut le contenu de dead letter

L'envoi du message se fait sur une nouvelle ligne en tapant soit Ctrl D soit un point suivi immédiatement de chariot retour. A ce moment là, une invite de copies à transmettre à d'autres utilisateurs vous est affichée (prompt Cc:), vous pouvez écrire une liste de "user@adress" et valider par chariot retour. Une liste vide est évidemment acceptée (pas d'autre utilisateur). Le message est alors envoyé. Dans le cas d'une adresse erronée, un courrier vous sera retourné par le "routeur" avec un message de non expédition.

8 - Impression

Après connexion sur la machine à laquelle l'imprimante est attachée :

lp [*options*] *fichier...fichier*

Pour imprimer un fichier postscript sur l'imprimante connectée à la machine ens13 ou un fichier ASCII sur l'imprimante connectée à la machine ens21.

enscript [*options*] *fichier...fichier*

Pour imprimer un fichier ASCII sur l'imprimante connectée à la machine ens13

9 - Avantages du terminal X-window

Pour certains logiciels, il n'y a même pas avantage mais obligation d'utilisation de l'environnement X-window avec quelquefois des compléments tels que des cartes spécifiques graphiques et leurs modules associés pour pouvoir les piloter. Par exemple pour des programmes de prédiction d'antigénicité des protéines qui sont des diagrammes de la valeur d'un paramètre en fonction de la position dans la séquence, il est plus facile de les visualiser dans des fenêtres graphiques par une représentation en deux dimensions. Tous les logiciels de modélisation moléculaires utilisent des représentations dites 3D etc....

En dehors de ces cas de nécessité, il est de toute façon plus agréable d'utiliser un environnement X-window pour les raisons suivantes :

- vous pourrez, à condition de respecter l'ouverture de fenêtres de la manière définie précédemment (page 5) utiliser chacune d'entre elles (fenêtre de type X-term) comme un shell et y lancer toute commande.
- par défaut une partie de texte affiché dans une fenêtre est gardé en mémoire et vous avez accès aux dernières lignes affichées (le nombre de lignes disponibles est ajustable)
- vous pourrez copier facilement du texte dans une fenêtre et l'insérer à l'endroit désiré (opération fastidieuse avec des éditeurs ligne ou page)

9.1 - Une fenêtre standard

Une fenêtre se présente comme une partie d'écran, rectangulaire avec une barre de titre où se trouvent à l'extrémité gauche une case et à l'extrémité droite deux cases, avec une barre de défilement vertical située à droite et avec une case située en bas et à droite de la fenêtre. Chacun de ces icônes spécifique permet une action particulière.

- La case de gauche à l'extrémité droite de la barre de titre vous permet de **fermer momentanément** cette fenêtre et par défaut un petit icône de la forme d'un écran d'ordinateur apparaît en bas à gauche de votre écran; celui-ci est transportable et par un double-clic dessus à l'aide du bouton de droite de la souris, la fenêtre réapparaît en l'état.
- Cliquer dans la case de gauche ou placer le curseur de la souris sur la barre de titre à l'exclusion des deux cases et ensuite cliquer sur le bouton de droite affiche un menu. Le seul menu que vous utiliserez est le menu "Fermer la fenêtre" qui permet de **fermer définitivement** la fenêtre. **Attention**, tous les processus lancés à partir de celle-ci ne sont plus accessibles. N'utilisez jamais cette action si des processus ont été lancés et ne sont pas terminés.
- Placer le curseur de la souris sur la barre de titre à l'exclusion des cases et cliquer sur le bouton de gauche ou milieu en le maintenant enfoncé permet de déplacer la fenêtre.
- Placer le curseur de la souris sur la case inférieure de la fenêtre et cliquer sur l'un des

boutons en le maintenant enfoncé permet de modifier la taille de la fenêtre.

9.2 - Une opération bien utile : la copie de texte et son insertion

Sélection de texte

Copier une partie de texte s'effectue à l'aide du bouton de **droite ou gauche** de la souris. Vous positionnez le curseur de la souris au début du texte, en maintenant le bouton de droite de la souris enfoncé, vous déplacez le curseur jusqu'à la fin du texte à sélectionner (celui-ci se noircit) en déplaçant la souris. Lorsque le texte est sélectionné vous relâchez le bouton. Le texte est mis en mémoire dans un tampon particulier et il est disponible pour être copié à l'endroit désiré.

Copie de texte

La recopie du texte se fait à l'aide de la souris au point d'insertion en cliquant sur le bouton du **milieu**. Vous avez deux cas possibles

- soit vous voulez insérer ce texte dans la ligne active de commande. Dans ce cas, placez le curseur souris sur la ligne et cliquez sur le bouton du milieu de la souris.

Attention : si vous avez dans le texte copié un retour chariot, la commande représentée par la ligne plus le texte que vous avez inséré va être lancé.

- soit vous voulez insérer ce texte dans un fichier édité à l'aide de vi : dans ce cas, déplacez le point d'insertion à l'aide des touches de déplacement à l'endroit désiré (***attention ce curseur n'a rien à voir avec celui de la souris***), commutez en mode insertion, placez le curseur de la souris sur la ligne et cliquez sur le bouton du milieu de la souris. Si vous êtes bien en mode insertion, le texte est inséré.

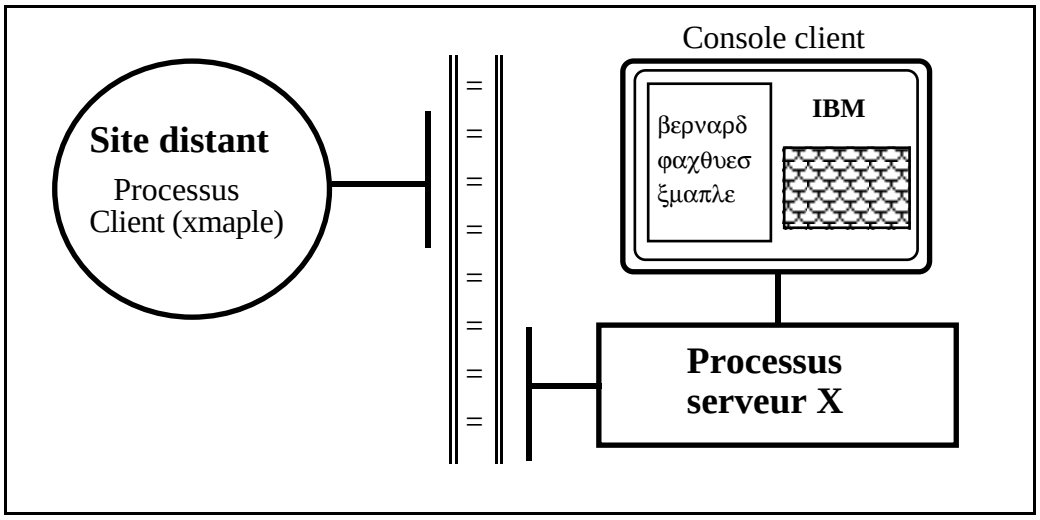
Un seul tampon est disponible pour effectuer ces opérations : ceci signifie qu'une nouvelle sélection va effacer l'ancien contenu et le remplacer par la sélection que vous êtes en train de faire.

10 - X-window et clients/serveurs en réseau

X est un serveur d'afffichage contrôlant l'écran, le clavier et la souris du terminal utilisé. Il est composé de trois parties :

- un serveur qui contrôle le terminal et ses périphériques
- des clients qui font des requêtes au serveur
- un moyen de communication entre client et serveur

Voici un schéma classique de fonctionnement de sessions avec le système X-window.



Deux cas différents peuvent se présenter selon que le processus client se déroulant sur le site distant est un processus demandant un terminal classique (X-term) ou un processus exigeant un environnement X-window

10.1 - Emulation d'un terminal X

Si les machines locales et distantes émulent des terminaux de même type dans le même mode, pas de problème. Dans un cas différent, la fenêtre locale dans laquelle le processus distant va agir, doit émuler le même type de terminal prédéfini par le système distant. Par exemple, j'utilise localement une station IBM dont, par défaut, le type aixterm est un terminal classique et le mode HFT, et je veux utliser le "gopher" de Polytechnique qui m'oblige d'avoir un terminal classique en mode vt100. Avant de réaliser une connexion par la commande telnet, je dois tenir compte de ces impératifs et réaliser l'opération 1 ou 2 :

1) Dans la fenêtre où je vais réaliser la connexion, je donne à la variable TERM la valeur adéquate par la commande :

Sous C-shell `setenv TERM vt100`

Sous un autre shell `TERM=vt100` puis `export TERM`

2) Je crée une fenêtre en utilisant une option qui va permettre l'émulation du mode vt100. Ceci s'obtient en ajoutant l'option v (spéccifique IBM) : aixterm -v -T goph -sb &

Attention : le changement de mode implique que des fichiers de correspondance clavier (Keyboard map) soient configurés correctement.
La valeur des variables d'environnement s'obtient par la commande **env**.

10.2 - Emulation d'un environnement X-window

J'ai besoin d'utiliser un logiciel (tournant sous X-window) implanté sur une machine à

laquelle j'ai accès par le réseau. Localement je vais lancer X-window, puis me connecter à la machine distante et faire éxécuter le proramme avec un affichage sur le terminal dépendant de la machine locale.

Lorsque l'environnement X-window est lancé par la commande "open xinit", un seul client est déclaré par défaut, c'est le terminal que j'utilise. Il faut donc que j'indique à la machine locale que d'autres clients sont possibles. Ensuite, une fois la connexion réalisée il faudra indiquer à la machine distante que l'affichage doit avoir lieu sur le terminal local et quel est le serveur qui s'occupe de la gestion des communications et de l'affichage.

xhost *options* [*adresse_client*]

Les option sont :

+ autorisation des clients définis par adresse_client
- suppression de l'autorisation

Si adresse_client est omis, alors l'action est appliquée pour tous les clients potentiels. Une fois l'autorisation donnée, je réalise la connexion avec la machine distante. Ensuite pour indiquer le gestionnaire et le terminal d'affichage, je donne à la variable d'environnement DISPLAY l'adresse I.P de la machine locale suivi de deux points et du numéro du terminal lié à la machine locale (terminal par défaut 0, à vérifier par la commande **env**)

```
% xhost +
% env
.........................
DISPLAY= unix:0                    (si  unix:n)
.........................
% tn ylu.cnrs-mrs.fr
.................                  Connexion classique
% setenv  DISPLAY  139.124.12.4:0        (alors :n)
%xmaple
```

Références

Dominique BOUILLET (1990) UNIX : Guide l'utilisateur. *Ellipses Int. Telecom.*

Henri GARETTA (1993) Utiliser UNIX *DESS IDC Faculté de Luminy*

Jean-Marie RIFLET (1993) La programmation sous UNIX *Ediscience International*

Manuel d'utilisation : peut être consulté en ligne par : **man** nom_commande
(appelle la documentation en ligne et liste les informations concernant la commande)