



Objectifs du TP : commandes minimales pour utiliser un ordinateur sous *Linux*, sans interface graphique, uniquement en ligne de commande. Donc :

- Comprendre l'arborescence de fichiers et connaître les commandes relatives à cette arborescence (*mkdir, cd, ls, pwd, ...*),
- Savoir éditer un fichier et utiliser quelques commandes de base sur les fichiers (*cat, cp, rm, ...*),
- Connaître quelques commandes sur les processus (*ps, kill*).
- Savoir installer des *packages* binaires et compiler ses propres programmes en binaire.

1. Qu'est-ce que Linux ?

Linux est un système d'exploitation libre, implémentation « *open source* » du système *Unix*.

Il est important de noter que connaître les commandes sous *Linux* permet de savoir utiliser tous les systèmes dits « *Unix-like* » (divers *Unix* commerciaux, *OpenBSD*, *FreeBSD*, etc.).

La part de marché de *Linux*, faible sur les ordinateurs personnels (~2 %), est élevée sur les serveurs (~60 % à 80 % *Linux* et *Unix-like*), les super-ordinateurs (100 %) et les systèmes embarqués (~70 %).

À la différence des systèmes « propriétaire » *Windows* (*Microsoft*) et *Mac OS X* (*Apple*), qui sont monolithiques, le côté « libre » de *Linux* favorise l'existence de différentes branches (« *forks* »), appelées « **distributions *Linux*** ». Ce sont des suites logicielles toutes basées sur ce qu'on appelle le « noyau *Linux* » (« *Linux kernel* »). Il existe des centaines de distributions, mais on peut les regrouper en quelques grandes familles dont le nom est donné par la distribution initiale (historiquement) :

- La famille *Debian*  , avec notamment les distributions *Ubuntu*  , ou encore *Linux Mint* 
- La famille *Red Hat*  , avec les distributions *Fedora* ,  *Centos* , 

Et des familles un peu moins importantes :

- *Slackware*  ,
- *Arch Linux*  ,
- etc...

Auxquelles on doit rajouter :

- Le système *Android*  , assez spécifique, mais qui est basé sur un noyau *Linux*.

Ce qui différencie toutes ces distributions ne se situe pas au niveau des commandes de base (en mode « texte », cf. §2), mais surtout au niveau des environnements graphiques et outils installés.

2. Le mode « ligne de commande » (appelé aussi mode « texte », « terminal » ou « console »)

Linux dispose d'un mode graphique, analogue à *Windows* et *Mac*, très complet et convivial. Cependant, il peut être nécessaire d'utiliser le mode « ligne de commande » (beaucoup d'utilisateurs expérimentés de *Linux* préfèrent d'ailleurs utiliser ce mode), pour plusieurs raisons :

- Le mode graphique est consommateur de ressources processeur et mémoire, ce qui est gênant pour des systèmes ayant des ressources limitées (carte d'un système embarqué, routeur,...) ou pour des serveurs fortement sollicités. Pour ce type de machines, il n'y a pas de mode graphique installé.
- La connexion en ligne de commande sur des machines distantes (par *telnet* ou *ssh*) est moins consommatrice en bande passante réseau qu'une connexion en mode graphique.
- Lors de certaines pannes logicielles (ex : « *freeze* » du système graphique), le mode « ligne de commande » est indispensable pour s'en sortir.
- Les commandes en mode « texte », grâce à leurs nombreuses options et à la possibilité de les chaîner en « *pipeline* », permettent une grande souplesse et sont plus polyvalentes (« *versatile* ») qu'une application graphique.

Syntaxe générale d'une commande : commande [options] paramètres

En général, les options (souvent une seule lettre) sont précédées du symbole « - » et peuvent être groupées. Les paramètres indiquent en général le(s) fichier(s) concerné(s).

Exemple de commande :

```
tar -xzvf fichier.tgz
```

avec :

Commande : tar

Options : x, z, v et f

Paramètre : fichier.tgz

Attention ! bien respecter la casse des caractères (c'est-à-dire les majuscules et minuscules).

Remarque : sous *Unix/Linux* l'interpréteur de commandes s'appelle le « *shell* » (coquille).

3. En pratique, pour démarrer :

- Quelle que soit la distribution *Linux* que vous utiliserez (*Ubuntu*, *Debian*, *Centos*, ...), vous commencerez par vous connecter avec votre nom d'utilisateur (*isen* par ex.) et votre mot de passe (c'est ce qu'on appelle le « *login* ») au début de votre session.
- Ensuite, vous allez ouvrir un « terminal » : c'est une simple fenêtre (on l'appelle aussi parfois « console ») qui permet de taper des commandes « en mode texte ». Pour la réalisation de ces TP sous *Linux*, il est interdit d'utiliser des outils graphiques, comme le « *File Manager* » (c'est dur, mais c'est comme ça qu'on apprend ...). Selon les distributions, pour ouvrir un terminal, il faut chercher dans le menu « Applications->Accessoires », ou « Applications->Outils Système »
- Dans le terminal, chaque ligne est précédée à gauche par une chaîne de caractères que l'on appelle le « **prompt** ». En général, le *prompt* est constitué du nom d'utilisateur, suivi de « @ », suivi du nom de la machine, puis du répertoire courant et de « \$ » (ou « # » si vous êtes *root*). Si vous tapez une commande, le *prompt* va se réafficher en début de ligne quand la commande sera terminée.
- Vous pouvez (éventuellement) modifier les caractéristiques d'affichage de votre terminal (police de caractères, couleur, taille, ...) à l'aide du menu situé au-dessus du terminal.

Remarque : Il y a une autre façon de travailler en mode texte : passer en « **terminal virtuel** » en tapant *Ctrl Alt F2* (en même temps les touches *Control*, *Alt* et *F2*). Vous vous retrouvez alors dans un vrai

environnement texte. Pour la plupart des distributions *Linux*, il y a 7 terminaux virtuels (touches *F1* à *F7*). Sous *Ubuntu* et autres *Debian*, *Fedora*<10 et *Centos* <6 les touches *F1* à *F6* font passer en mode texte, *F7* en mode graphique. Sous *Fedora* ≥10 et *Centos* 6, *F2* à *F7* font passer en mode texte, *F1* en mode graphique. Pour revenir à votre terminal graphique de départ : *Ctrl Alt F7* (ou *Ctrl Alt F1* pour les versions récentes de *Fedora* et *Centos*).

4. Quelques généralités sur la ligne de commande

- **Facilités d'édition** de la ligne de commande : en cours de frappe d'une ligne de commande, on dispose de plusieurs méthodes pour faciliter l'écriture :
 - Les touches flèches « ← » et « → » : elles permettent de se déplacer dans la ligne,
 - Les touches « ←- » (<Backspace>) et « Suppr » () : elles permettent de supprimer respectivement le caractère précédant le curseur et à l'endroit du curseur.
 - Le « *copy-paste* » à la souris : si vous avez ouvert un terminal « texte » sous le mode graphique, vous pouvez sélectionner à la souris un texte quelconque. Un clic sur le bouton du milieu (molette) - ou le bouton droit pour une souris à 2 boutons - recopie ce texte sur la ligne de commande, même dans un autre terminal. Vous pouvez aussi utiliser les raccourcis clavier *Shift Ctrl c* (*copy*) et *Shift Ctrl v* (*paste*).
 - La « **complétion** » (ou « complètement ») de commande :
Lorsqu'on tape une ligne de commande incomplète, puis la touche <TAB>, le *shell* cherche à compléter le mot avec un nom de fichier ou de commande, suivant le contexte. S'il y a plusieurs propositions, il y a attente d'un complément d'information de la part de l'utilisateur (avec un son « *bip* » sur certaines installations). Ex. sur la commande *cat* dont vous verrez l'explication ultérieurement (ici, elle produit l'affichage du contenu du fichier *.bashrc*) : en rouge ce que vous tapez (<TAB>=  touche), en noir ce qui s'affiche dans le terminal.
\$ **cat** **.b**<TAB>
\$ cat .bash
\$ **cat** **.bashr**<TAB>
\$ cat .bashrc
- **Rappel des commandes** précédentes :

On peut rappeler les commandes précédentes à l'aide de la touche « ↑ », puis revenir vers les commandes les plus récentes à l'aide de la touche « ↓ ».

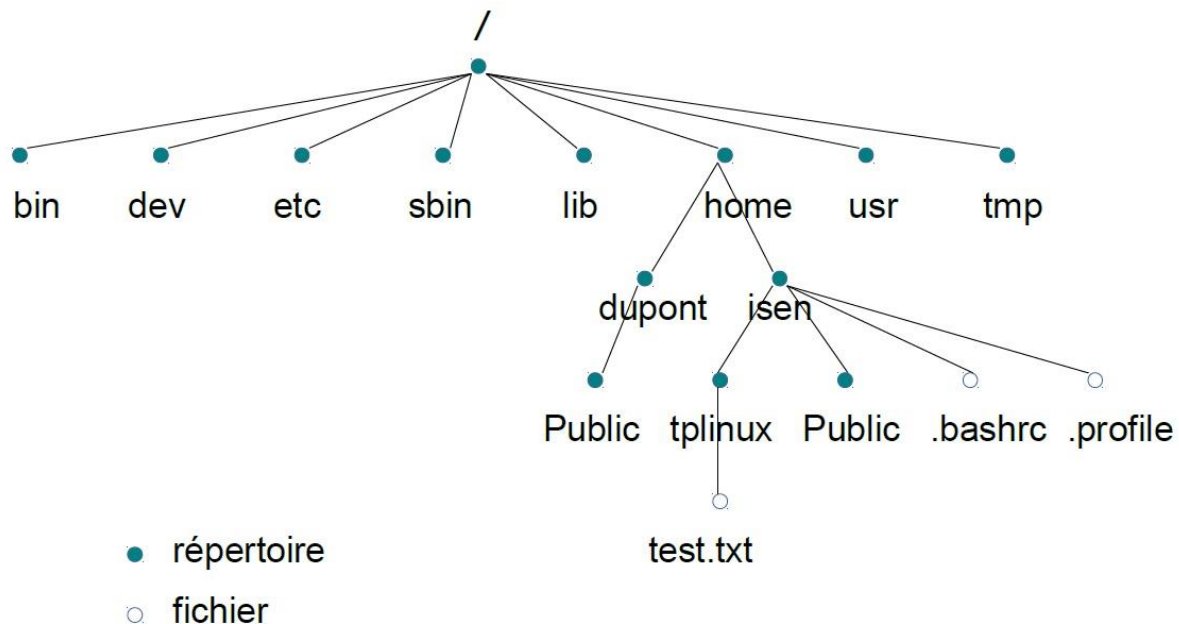
5. Pour arrêter votre machine : (pas tout de suite, quand vous aurez fini...)

- On n'arrête jamais une machine en appuyant sur le bouton  (marche/arrêt), même pour une machine virtuelle : le risque est d'arrêter le système dans un état incohérent et de ne pas pouvoir redémarrer.
- Pour arrêter en terminal graphique, il faut chercher dans le menu (variable selon les distributions, mais souvent en haut à droite) un item « Éteindre ».
- Si vous êtes en terminal purement « texte », il faut taper la commande : *sudo poweroff*
Pour pouvoir exécuter cette commande, on vous demandera votre mot de passe.

6. L'arborescence de fichiers et les commandes liées

Le système de fichiers de *Linux* présente une structure arborescente. En *Linux*, les dossiers (cf. *Windows*) sont appelés « répertoires » (*directories*). La racine de l'arborescence est toujours le

répertoire « / » (*root directory*). Ex. d'arborescence de fichiers :



En général, sous *Linux*, le répertoire personnel (*home directory*) d'un utilisateur se trouve sous « /home ». Ex : « /home/isen » pour l'utilisateur « isen ». Votre répertoire personnel peut être appelé « ~ » (*tilde*) dans les commandes.

Dans un terminal, on est toujours positionné à tout moment dans un répertoire (après un *login*, on est automatiquement dans le « *home directory* »). Le répertoire dans lequel on se trouve est appelé « répertoire courant » (*working directory* ou *current directory*). Il peut être appelé « . » (point) dans les commandes.

Le répertoire au-dessus du répertoire courant (répertoire parent) peut être appelé « .. » (point point) dans les commandes.

Les répertoires au-dessous du répertoire courant sont appelés sous-répertoires (*subdirectories*).

Deux répertoires ayant le même répertoire parent sont appelés répertoires « frères » (*sibling directories*) : dans l'exemple d'arborescence ci-dessus, les répertoires *tlinux* et *Public* sont frères.

Nom d'un fichier ou d'un répertoire :

Chaque fichier (*file*) ou répertoire (*directory*) est identifié de façon complète par l'expression du chemin (*path*) qu'il faut parcourir depuis le répertoire racine pour y arriver. On distingue :

- Le nom du fichier (*filename*) au sens strict : c'est le nom simple sans le chemin. Dans l'exemple d'arborescence ci-dessus, on trouve les noms de fichiers suivants : *test.txt*, *.bashrc* et *.profile*. Sous *Linux*, il n'y a **aucune obligation d'extension à la fin d'un nom de fichier** (contrairement à l'habitude sous *Windows*) : par ex. on peut trouver un fichier *test1.txt* ou *test1* ou *test1.texte.sauv*.
- Le nom absolu (=chemin absolu) : c'est le chemin complet depuis « / ». Dans l'exemple d'arborescence ci-dessus, le fichier *test.txt* a pour nom absolu « /home/isen/tlinux/test.txt ».
- Le nom relatif (=chemin relatif) : c'est le chemin depuis le répertoire courant. Dans l'exemple d'arborescence ci-dessus, le fichier *test.txt* a pour nom relatif :

« test.txt » ou « ./test.txt »	si on est dans le répertoire courant « /home/isen/tlinux »,
« tlinux/test.txt »	- - - - - « /home/isen »,
"../tlinux/test.txt"	- - - - - « /home/isen/Public »,
"../../isen/tlinux/test.txt"	- - - - - « /home/dupont/Public »,

Caractères autorisés dans les noms de fichiers :

Formellement, *Linux* accepte tous les caractères dans les noms de fichiers et de répertoires, sauf « / » et « \0 » (caractère NUL, fin de chaîne en C).

En fait, pour éviter des problèmes dans les commandes *shell*, il vaut mieux éviter tous les méta-caractères (*?:[]">|(){}&'!\;%#) et les caractères de contrôle. Il est aussi préférable d'éviter les espaces (ou « blanc »), surtout en fin de nom (ils ne sont pas visibles quand le nom s'affiche), et les « - » en début de nom (le nom de fichier pourrait être pris pour une option de la commande).

Si un de ces caractères apparaît dans un nom de fichier, il faut le « protéger » (l'« échapper ») par un « \ » (*antislash* ou *backslash*) ou mettre des « quotes » autour du nom. Par exemple, il faut écrire le nom de fichier « cinq et six<sept » :

cinq\ et\ six\<sept	(avec échappement)
'cinq et six<sept'	(entre apostrophes ou « <i>singles quotes</i> »)
"cinq et six<sept"	(entre guillemets ou « <i>doubles quotes</i> »)

Fichiers et répertoires cachés :

Certains fichiers et répertoires ont un nom commençant par « . » (point). Ils sont en général situés dans le « *home directory* » de chaque utilisateur. Ce sont des fichiers de configuration personnels (ou des répertoires contenant des fichiers de configuration). Dans l'exemple d'arborescence ci-dessus, on trouve les fichiers cachés *.bashrc* et *.profile*.

On dit qu'ils sont cachés, car ils ne sont pas visibles par la commande « *ls* » simple (cf. ci-dessous).

Substitution de « * » dans les noms de fichiers et répertoires :

Quand on donne des noms de fichiers (ou des chemins), *Linux* permet de substituer une partie du nom par le « méta-caractère » *, qui correspond à n'importe quelle chaîne de caractères, y compris la chaîne vide.

Les commandes pour manipuler l'arborescence :

Dans ce TP, les commandes indiquées par une ★ doivent être absolument **mémorisées**. Ce sont des commandes que tout « linuxien » doit connaître sans se référer à la documentation.

Remarque : dans les exemples donnés ci-dessous dans le texte du TP, un « \$ » sera affiché en début de ligne pour montrer qu'il s'agit d'une commande à taper.

Les commandes suivantes servent à se positionner et à se déplacer dans l'arborescence, à créer et détruire des répertoires ou à en lister le contenu.

- La commande ***pwd*** (*print working directory*) : affiche le nom absolu du répertoire courant.

★ \$ pwd

- La commande ***mkdir*** (*make directory*) : crée un nouveau sous-répertoire du répertoire courant. Ex :

★ \$ mkdir tplinux

- La commande ***ls*** (*list*) : liste le contenu d'un répertoire. Il existe beaucoup d'options, nous allons voir les plus courantes.

— « *ls* » seul affiche juste le nom des fichiers et des sous-répertoires du répertoire courant (sauf ceux commençant par « . »).

★ \$ ls

- « *ls -l* » idem « *ls* » seul, mais affiche toutes les caractéristiques des fichiers et des sous-répertoires (droits, propriétaire, taille, date,...). Un alias (raccourci) « *ll* » existe dans beaucoup de distributions (mais pas toutes).



```
$ ls -l
$ ll
```

- « *ls -a* » affiche en plus les fichiers et sous-répertoires commençant par « . ».



```
$ ls -a
```

- « *ls -al* » combine « *ls -a* » et « *ls -l* ».



```
$ ls -al
```

- « *ls rep* » affiche tous les fichiers et sous-répertoires du sous-répertoire *rep*. Ex :

```
$ ls tplinux
```

- « *ls *.txt* » affiche tous les fichiers avec l'extension « *.txt* ».

```
$ ls *.txt
```

- La commande ***cd*** (*change directory*) : change le répertoire courant, c'est-à-dire vous permet de vous déplacer dans l'arborescence. La commande « *cd* » seule vous positionne dans le « *home directory* » (équivalent à « *cd ~* »). Ex :



```
$ cd
$ cd ..
$ cd isen/tplinux
$ cd ../Public
```

- la commande ***rmdir*** (*remove directory*) : efface un répertoire, mais uniquement s'il est vide. Cette commande nécessite donc auparavant d'effacer les fichiers contenus dans le répertoire. Exemples :

« *rmdir rep* » supprime le répertoire *rep*. Ex :



```
$ rmdir tplinux
```

7. Les commandes de manipulation des fichiers

Création/édition d'un fichier :

- La commande ***touch*** crée un fichier vide dans le répertoire courant (ou change sa date de dernière mise à jour s'il existe déjà). Ex :

```
$ touch test.txt
```

- **Éditer un fichier**

Un éditeur de texte est un logiciel qui permet de visualiser le contenu d'un fichier texte, mais aussi de le modifier et éventuellement de le créer. Il en existe de très nombreux.

Un éditeur de base sous *Linux* est ***gedit***, mais nous ne l'utiliserons pas ici, car c'est un éditeur graphique (on se rappelle que ce TP doit vous apprendre à travailler en « mode texte » ;-). C'est l'éditeur graphique le plus simple d'utilisation. Il y en a de plus complets, par ex. *atom*.

Il existe plusieurs éditeurs en « mode texte » sous *Linux* : les plus connus sont *vi*, *nano* et *emacs*.

Dans ce TP nous utiliserons *nano* car il est plus simple d'utilisation que les autres. Dans un autre TP nous apprendrons à utiliser *vi* ou *vim* (*Vi IMproved*), qui est plus puissant que *nano* et toujours présent dans toutes les distributions de *Linux*.

```
$ nano test.txt
```

Les commandes de *nano* sont indiquées sur les lignes d'en bas de l'éditeur. Les principales sont *Ctrl o* (^O) pour sauver ce qu'on vient de taper et *Ctrl x* (^X) pour sortir de l'éditeur.

Ouvrez un fichier avec *nano* et tapez quelques lignes de texte quelconque, puis sauvez et sortez.

Affichage à l'écran du contenu d'un fichier :

- La commande **cat** : affiche à l'écran le contenu du(des) fichier(s) indiqué(s). Ex : ici on affiche le contenu du fichier *test.txt* :

★

```
$ cat test.txt
```

- Les commandes **more** et **less** sont très similaires : elles servent à visualiser un fichier page par page. *less* a plus d'options que *more*, mais leur fonctionnement basique est le même : pour afficher la page suivante, on appuie sur la touche « barre d'espace » (la touche « entrée » ne fait qu'afficher la ligne suivante), pour revenir à la page précédente, on appuie sur la touche « b » (*back*). On tape « q » pour quitter. Ex (afficher le contenu du fichier */etc/services*) :

★

```
$ more /etc/services
$ less /etc/services
```

Recopie, renommage et déplacement d'un fichier :

- La commande **cp (copy)** : copie un fichier source vers un fichier destination. Deux variantes :
cp fic_source fic_dest
cp fics_source rep_dest (recopie des fichiers vers un autre répertoire).

Exemples :

- « *cp fic_source fic_dest* » copie le fichier *fic_source* vers le fichier *fic_dest*. Ex (vous devez auparavant créer le répertoire TP1 pour que ces commandes fonctionnent) :

★

```
$ cp test.txt test2.txt
$ cp test.txt TP1/test.txt
```

- « *cp fic_source rep_dest* » copie le fichier *fic_source* vers le répertoire *rep_dest*. Ex : (cette commande fait la même chose que la seconde commande ci-dessus).

★

```
$ cp test.txt TP1
```

- « *cp fics_source rep_dest* » copie les fichiers *fics_source* vers le répertoire *rep_dest*. Ex :

★

```
$ cp *.txt TP1
$ cp * /tmp
```

- La commande **mv (move)** : renomme un fichier (ou répertoire) ou déplace un(des) fichier(s) vers un répertoire destination. La forme correspondant au renommage est :

mv fic_ou_rep_source fic_ou_rep_dest.

La forme correspondant au déplacement est :

mv fics_source rep_dest

Exemples (vous devez auparavant créer un répertoire TP2, « frère » de TP1, pour que ces commandes fonctionnent) :

- « *mv fic_source fic_dest* » renomme le fichier *fic_source* en *fic_dest*.

Ex :



```
$ mv test.txt test1.txt
```

- « *mv rep_source rep_dest* » renomme le répertoire *rep_source* en *rep_dest* (*rep_dest* ne doit pas être un répertoire déjà existant).

Ex :

```
$ mv TP1 TP0
```

- « *mv fic_source rep_dest* » déplace le fichier *fic_source* vers le répertoire *rep_dest*.

Ex :



```
$ mv test1.txt ../TP2
```

- « *mv fics_source rep_dest* » déplace les fichiers *fics_source* vers le répertoire *rep_dest*.

Ex :

```
$ mv *.txt ../TP2
```

- La commande **rm (remove)** : efface un(des) fichier(s). La forme générale est : **rm fic**. Une seconde forme est : **rm fic1 fic2 fic3 etc.**

Attention ! Danger : les suppressions sont définitives (pas de corbeille !). Exemples :

- « *rm fic* » supprime le fichier *fic*. Ex :



```
$ rm test2.txt
```

- « *rm fics* » supprime les fichiers *fics* (listés un par un ou avec « * »). Ex :

```
$ rm test.txt test2.txt
```

```
$ rm *.txt
```

Exercices :

Créez sous votre « *home directory* » un répertoire *tplinux* (s'il n'existe pas déjà) et allez dans ce répertoire. Téléchargez le fichier *Tp1linux.tar.gz* dans le répertoire *tplinux*. Vous le faites en ligne de commande en tapant :

```
$ wget --no-check-certificate "http://litlurl.net/674f26"
```

(Pour une raison inconnue, il arrive que *wget* renomme ce fichier automatiquement en « *674f26* », si c'est le cas chez vous, vous pouvez le renommer en tapant la commande *mv 674f26 Tp1linux.tar.gz*)

Un fichier « *tar* » contient, de façon compressée, toute une partie d'arborescence. Assurez-vous que vous êtes dans le répertoire *tplinux* et décompressez ce fichier « *tar* » à l'aide de la commande suivante :

```
$ tar -xvf Tp1linux.tar.gz
```

À l'aide des commandes vues précédemment :

- Listez ce qu'il y a maintenant dans le répertoire *tplinux*.
- Sur une feuille de papier, représentez l'arborescence existante sous *tplinux* (soit en vous déplaçant dans les différents répertoires, soit à l'aide de la commande *du*).
- Déplacez-vous dans le répertoire *Tp1*.
- Supprimez en une seule commande les fichiers *fic_destruc1* et *fic_destruc2*.
- Créez (sous *Tp1*) deux répertoires : *programme* et *donnees*.

- Déplacez, en une seule commande, dans le répertoire *programme* tous les fichiers suffixés par « .c ».
- Déplacez, en une seule commande, dans le répertoire *donnees* tous les fichiers suffixés par « .dat » ou « .txt ».
- Détruisez le répertoire *REP_INUTIL*.
- Sur une feuille de papier, représentez la nouvelle arborescence existante sous *tplinux*.
- Allez dans le répertoire *programme*.
- Affichez le fichier *liste.c* (avec *cat*, *more*, *less*).
- Créez une copie du fichier *liste.c* dans le répertoire parent *Tp1* (appelez la copie *liste2.c*).
- Modifiez (ajoutez ou enlevez un mot) à l'éditeur une ligne quelconque du fichier *liste2.c*.
- Trouvez un moyen, sans ré-afficher *liste.c* et *liste2.c*, de détecter qu'il y a une différence entre les 2 fichiers.

8. Gestion des processus

Notion de processus (*process*) :

Un processus est un programme en cours d'exécution. Il est toujours lancé par un utilisateur (personne physique ou entité « système »).

Un processus est identifié par un numéro unique (*Process Id* ou **PID**). Les *PID* sont attribués par le système. Le premier processus lancé après le démarrage du système *Linux* est le programme *init*, lancé par *root*, qui porte le *PID* 1.

Tous les processus sont lancés à partir d'un processus parent. Par exemple, les commandes ou programmes que vous lancez dans un terminal sont des processus fils d'un processus *bash*, lui-même fils de ce terminal. Le *PPID* (**P**arent **P**rocess **I**d) d'un processus est le *PID* de son processus parent.

Lister les processus :

La commande **ps** (*process status*) : liste les processus en cours d'exécution. Il existe beaucoup d'options, nous allons voir les plus courantes.

- « *ps* » seul affiche juste les processus qui s'exécutent dans le terminal, avec quelques informations.

```
$ ps
```

- « *ps -ef* » affiche tous les processus de la machine (e=*every*) avec beaucoup d'informations (f=*full*). C'est la commande dont il faut se rappeler.



```
$ ps -ef
```

- « *ps -u isen* » affiche tous les processus lancés par l'utilisateur *isen*.

```
$ ps -u isen
```

Lancer un processus en arrière-plan :

Un processus peut s'exécuter en « premier plan » (*foreground*), c'est-à-dire que vous « n'avez plus la main » dans le terminal (le *prompt* n'est pas réaffiché tant que le processus n'est pas terminé), ou en « arrière-plan » (*background*, dit aussi en tâche de fond), c'est-à-dire qu'il vous « rend la main » dans le terminal. C'est intéressant surtout pour les applications graphiques.

Pour lancer un processus en arrière-plan, on fait suivre la commande du symbole **&**. Cependant, un processus qui a été lancé en premier plan (sans **&**), peut être ramené en arrière-plan (*background*). Il faut pour cela taper <Ctrl-Z> pour suspendre son exécution, puis *bg*.

Ex : lancez l'application *xeyes* en tâche de fond. Vous avez la main dans le terminal (vous pouvez taper des commandes, par ex. *ps*).

Arrêtez *xeyes* en cliquant sur le symbole d'arrêt () . Relancez *xeyes* en premier plan, puis faites-le retourner en arrière-plan :



```
$ xeyes &  
$ ps  
$ xeyes  
$ <Ctrl-Z>  
$ bg
```

Arrêter (« tuer ») un processus :

Un processus qui a été lancé en premier plan depuis un terminal peut être arrêté (« tué ») facilement en tapant **<Ctrl-C>** dans le terminal.

C'est plus difficile pour un processus s'exécutant en arrière-plan : il faut lui envoyer un signal d'arrêt, à l'aide de la commande *kill*. Il faut pour cela connaître son PID (à l'aide de la commande *ps*).

La commande *kill* :

Elle ne peut être effective que sur les processus qui vous appartiennent (c'est-à-dire que vous avez lancé vous-mêmes). Elle peut cibler plusieurs processus.

kill [-signal] pid [pid2]...

Le principal **signal** à connaître est 9 : il demande au processus de s'arrêter immédiatement et ne peut être bloqué. Par défaut, si aucun signal n'est fourni, c'est le signal 15 qui est envoyé : il demande aussi l'arrêt, mais peut être mis en attente par le processus s'il est en train d'effectuer un traitement.

Exemple (on suppose que le PID de *xeyes* est 5225) :



```
$ xeyes  
$ <Ctrl-C>  
$ xeyes &  
$ ps  
$ kill -9 5225
```

9. Notions d'utilisateur et de groupe

- Toute entité (personne physique ou programme) devant interagir avec un système *Linux/Unix* doit être authentifiée sur un ordinateur en tant qu'utilisateur ou **user**. Un utilisateur est reconnu par un nom unique et un numéro unique (*User Id* ou *UID*).
- Il y a toujours sur un système *Linux/Unix* un utilisateur particulier dit « super-utilisateur » (*superuser*) dont, par convention, le nom est **root** et l'*UID* est 0. Il a tous les droits sur les fichiers et les programmes.
- Un utilisateur appartient à au moins un **groupe** (*group*). Les groupes servent à rassembler des utilisateurs afin de leur attribuer des droits communs. Un groupe porte un nom et un numéro unique (*Group Id* ou *GID*).

Le groupe principal d'un utilisateur se nomme « groupe primaire ». Par défaut, sous *Linux*, quand un utilisateur est créé, un groupe de même nom est aussi créé, qui ne comporte que cet utilisateur. Par exemple, vous êtes l'utilisateur *isen*, dans le groupe *isen*.

Un utilisateur peut être membre d'autres groupes, appelés « groupes secondaires », qui

correspondent souvent à des logiciels sur lesquels vous avez des droits particuliers. Un de ces groupes secondaires s'appelle (selon les distributions) *admin*, *wheel* ou *sudo* : il vous donne des droits d'administrateur sur la machine (cf. § suivant).

Par convention, le groupe primaire du « super-utilisateur » est *root* et son *GID* est 0.

- Commande pour connaître son UID et le GID de son(ses) groupe(s) :

★ `$ id`

Pour connaître les groupes dont vous faites partie :

`$ groups`

La commande *sudo* :

La commande ***sudo*** permet de changer d'identité (généralement pour passer « super-utilisateur ») uniquement pendant le temps d'exécution d'une commande. Le nom *sudo* est une abréviation de « ***substitute user do*** », c'est-à-dire « exécuter en se substituant à l'utilisateur ».

Pour être autorisé à lancer une commande avec *sudo*, un utilisateur doit faire partie d'un groupe autorisé (selon les distributions : *admin*, *adm*, *sudo*, *wheel*,...) ou être autorisé à titre personnel. On lui demandera un mot de passe (**Attention !** le sien et non pas celui de *root*) ; ce mot de passe ne sera pas redemandé si on relance une commande avec *sudo* pendant un certain laps de temps (5 à 15 minutes selon la distribution *Linux*).

Les utilisateurs autorisés à utiliser la commande *sudo* sont appelés *sudoers* (dans la plupart des distributions *Linux*, vous êtes mis automatiquement dans les « *sudoers* » quand vous êtes le premier utilisateur créé à l'installation du système sur la machine).

La commande ci-dessous consiste à vérifier si vous (utilisateur *isen* par ex.) êtes listé dans les « *sudoers* ». Elle liste les commandes que vous pouvez exécuter en *sudo* (*ALL* pour toutes).

`$ sudo -l`

10. L'installation d'applications (paquets ou « packages » binaires)

Nous allons voir, succinctement, l'installation d'applications en ligne de commande. Les commandes sont différentes selon les distributions. Dans les distributions *Debian* (*Ubuntu*), on utilise la commande ***apt-get***, dans les distributions *Red Hat* (*Centos*, *Fedora*), c'est ***yum***.

Dans les commandes indiquées ci-dessous, vous exécuterez la commande adaptée (*apt-get* ou *yum*) selon le système dans lequel vous êtes. Toutes ces commandes sont, *a priori*, à exécuter en *sudo*.

- Mise à jour de la liste des paquets disponibles : c'est automatique dans les distributions *Red Hat*. En *Debian* :

`$ sudo apt-get update`

- Installation d'une application. Ex. *sl* :

`$ sudo yum install sl`
`$ sudo apt-get install sl`

- Test du fonctionnement et de la documentation de l'application installée :

`$ sl`
`$ man sl`

- Désinstallation d'une application. Ex. *sl* :

```
$ sudo yum remove sl
$ sudo apt-get purge sl
```

11. La compilation de vos applications (sources C)

Nous allons voir, comment compiler vos propres applications, en prenant l'exemple d'un programme C. Le logiciel qui transforme vos fichiers de programme (fichiers sources) en langage machine binaire est appelé un **compilateur**. Le compilateur C généralement installé sous *Linux* est **gcc** (*GNU C Compiler*).

Tout d'abord vérifier que la commande *gcc* est bien installée : `which gcc`
Sinon il faut installer *gcc* (cf. §10 ci-dessus).

Compilation d'un simple fichier source C

Une fois *gcc* disponible, voici les différentes phases à suivre :

- Créez un répertoire *tpc* et allez dans ce répertoire.
- Vous allez partir d'un simple programme C que vous devez taper à l'éditeur :

```
#include <stdio.h>
int main()
{
    char nom[50] ;
    printf ("Tapez votre nom : ") ;
    fgets(nom, 50, stdin) ;
    printf (" Bonjour %s\n ", nom) ;
    return 0;
}
```

- Sauvez votre fichier sous le nom *bonjour.c*
- Compilez le programme :

★

```
$ gcc bonjour.c -o bonjour
```

L'option `-o` indique au compilateur de générer un fichier exécutable s'appelant *bonjour*. Si on ne met pas l'option `-o`, le nom par défaut de l'exécutable sera **a.out**.

- Exécutez le programme :

```
$ ./bonjour
```