# TP - Introduction à l'utilisation de LINUX

Une feuille de TD sera proposée pour chaque semaine, un peu adaptée pour la séance de TP. Si vous n'avez pas atteint la section *renforcements* avant la fin des TD/TPs de la semaine, vous devez les terminer par un travail à la maison, et éventuellement poser vos questions à la séance suivante.

**Objectif(s)**

- ✯ Prendre en main l'utilisation du Terminal sous Linux
- ✯ Être capable d'éditer un programme C, le compiler et l'exécuter.
- ✯ Découvrir quelques balises de printf.

N'hésitez pas à prendre des notes sur ce premier TP, vous utiliserez souvent ces commandes, et nous ne reviendrons pas dessus !

## Prise en main de linux et du terminal

Si votre machine a démarré sous *windows*, redémarrez-là et sélectionnez *ubuntu*. L'interface utilisateur par défaut de ce système d'exploitation ressemble beaucoup aux interfaces que vous connaissez, et vous devriez vous y retrouver. La touche "windows" vous permet d'ouvrir le menu principal, vous reconnaitrez l'icone de *firefox* pour naviguer sur internet. L'icône représentant un porte document est l'explorateur de fichiers, et vous permet de naviguer dans votre arborescence.

Dans le menu à gauche, vous pouvez trouver un élément appelé terminal ou console (ou xterm). Vous pouvez aussi ouvrir le terminal avec la séquence CTRL+ALT+T. Le terminal est un outil de communication direct avec la machine. Bien que peu convivial, il peut s'avérer très efficace. Vous allez utiliser le terminal comme interface de communication lors de la compilation, autant le prendre en main.

Nous allons voir comment utiliser le terminal.

1. Observez d'abord le texte qui est affiché dans le terminal. On l'appelle *invite de commande*. Vous y voyez votre numéro d'étudiant (login), suivi d'un `\@`, suivi du nom de la machine, et de `:`. Le $\sim$ qui suit vous indique le répertoire courant, $\sim$ étant utilisé pour désigner votre répertoire personnel. Le $ final est juste là pour conclure l'invite.
2. Utilisez la commande `ls`. Elle vous permet de lister le contenu du répertoire courant. Si vous voulez plus de détails sur les fichiers listés, vous pouvez utiliser la commande `ls -l` (bien respecter les positions des espaces).
3. La commande `mkdir` vous permet de créer un répertoire.
   Créez un répertoire *toto* avec la commande `mkdir toto`. Avec `ls`, vérifiez que le répertoire a bien été créé. Créez de la même façon un repertoire *titi* et un répertoire *tata*.
4. La commande `cd` vous permet de changer de répertoire. Entrez dans le répertoire *toto* avec `cd toto` et vérifiez qu'il est vide. Remarquez que votre invite de commande vous indique maintenant que vous vous trouvez dans le répertoire $\sim$/toto.
5. Le répertoire parent du répertoire courant est le répertoire `..`. Utilisez `cd ..` pour retourner dans votre répertoire principal. Entrez maintenant dans le répertoire *titi*. Constatez à nouveau qu'il est vide et le changement de l'invite. Sautez maintenant directement dans le répertoire *tata* avec la commande `cd ../tata`. Vous pouvez ainsi composer des adresses de répertoire pour vous déplacer plus rapidement dans les dossiers. Sautez maintenant dans le répertoire *titi*.
6. Il peut paraître fastidieux de taper toutes les commandes. De plus, il est parfois difficile de se souvenir du nom complet d'un répertoire. Il existe une touche *magique*, la touche tabulation. Elle permet de faire de l'autocomplétion ou d'avoir une liste des possibilités. Testez son utilisation pour auto compléter les noms de répertoire, en retentant les déplacements précédents. L'autocompletion fonctionne aussi avec les noms de commandes (mais n'est pas très utile pour les commandes à 2 lettres).

7. La commande `mv` vous permet de déplacer des fichiers ou des répertoires. Pour cela, vous allez avoir besoin d'adresses composées. Placez vous dans le répertoire *toto* et déplacer le répertoire *titi* dans celui-ci grâce à la commande `mv ../titi ./` (dans laquelle `.` désigne le répertoire courant). Observez que le répertoire titi est bien dans votre emplacement courant, et allez vérifier qu'il ne se trouve plus dans votre répertoire personnel ($\sim$).
8. L'outil que vous allez utiliser initialement pour programmer est l'éditeur de texte de base de *Gnome*, qui s'appelle `gedit`. Tapez la commande `gedit programme.c` pour créer un fichier *programme.c* et le modifier. Vous perdez la main du terminal (qui attend que vous ayez fini d'utiliser gedit pour reprendre) et une fenêtre d'édition s'ouvre. Lorsque vous fermez la fenêtre d'édition, vous reprenez la main sur le terminal. Vous pouvez aussi garder la main dans le terminal en tapant `gedit` $\hookrightarrow$ `programme.c &` (avec l'ajout d'un & commercial).
9. Vous pouvez enfin tester la commande de suppression `rm`, qui vous permet de supprimer un fichier. Pour supprimer un répertoire et tout ce qu'il contient, vous pouvez utiliser `rm -r` (pour récursivement).
10. Un autre outil d'édition, plus riche et développé que *gedit*, s'appelle *geany*. Lancez cet autre éditeur dans la console, vous pourrez utiliser l'un ou l'autre au choix, ou même un autre éditeur. Profitez en pour tester l'autocompletion sur les noms de commande.
11. Vous pouvez aussi trouver ces deux éditeurs dans le menu d'accès aux différents programme d'Ubuntu. Fouillez un peu pour lancer *gedit* et *geany* via l'interface graphique d'Ubuntu. Vous noterez qu'en lançant les programmes ainsi, vous n'avez pas de problématique de perte de main dans la console, comme en utilisant le & commercial.
12. Dans le terminal, vous ne savez sans doute pas encore faire un copier-coller. En fait, c'est possible en selectionnant une partie de texte et en cliquant sur le bouton du milieu de la souris (la molette). Testez.
13. Les flèches vers le haut et le bas peuvent s'avérer très utiles aussi. Testez ces touches et déduisez leur usage. Avec toutes ces astuces (tabulation, bouton du milieu, flèches), vous devriez devenir très habiles avec un terminal, sous réserve d'un peu d'entraînement.

# Disque Campus

Les machines du campus sont configurées avec un disque distant, auquel vous avez accès depuis n'importe quelle machine, et même depuis chez vous via le cloud. Ce disque distant s'appelle *CAMPUS*, il se trouve comme un répertoire de votre *home*.

14. Depuis votre console, ouvrez le disque CAMPUS. Vous devriez y retrouver un dosssier du nom de votre login. Entrez dans le dossier et listez les fichiers et dossiers qui s'y trouvent (vous aurez peut-être déjà des fichiers que vous auriez créé dans d'autres cours).
15. Créez[1] un fichier `temoin.txt` dans lequel vous pouvez écrire la date du jour. N'oubliez pas de sauvegarder. Ce fichier sera désormais accessible depuis n'importe quelle machine.
16. Connectez-vous sur `https://bureau-distant.unicaen.fr` avec vos identifiants de session. Vous devriez y retrouvez votre fichier `temoin.txt` (dans le disque *Z* ou *calebasse*).
17. Vous pouvez aussi accéder au bureau distant via un client lourd (i.e. un logiciel installé sur la machine). Lancez l'application Vmware Horizon client (une icone verte dans le menu à gauche), vous pouvez aussi vous connecter ainsi au bureau distant.

Dans toutes les séances, vous pouvez travailler sur le disque local pour éviter les problèmes de performances du réseau (recommandé), mais pensez avant de quitter votre station de travail à copier les fichiers sources qui vous intéressent sur votre disque CAMPUS afin de les sauvegarder pour la suite. **Tout le contenu hors CAMPUS est effacé au redémarrage de la machine !**

1. avec gedit ou geany, pas avec mkdir

# Compilation d'un programme C

18. Dans un fichier `bonjour.c`, avec l'extension `.c`, écrivez le programme *Hello world* suivant :

```
#include <stdio.h>
int main(void){
  printf("Hello world\n");
  return 0;
}
```

On notera le caractère `\n` qui sert de retour à la ligne.

19. Sauvegarder le fichier, et compilez le avec `gcc -Wall bonjour.c -o bonjour`. Ceci crée un fichier `bonjour` qui contient votre programme.
20. Lancez votre programme en tapant `./bonjour` dans le terminal. Vous devriez voir apparaître "Hello world" dans la console.
21. Dans la ligne de compilation, comment faire pour que le fichier dans lequel le programme enregistré ne s'appelle pas `bonjour` [2] ? Quel est le nom de fichier utilisé par défaut ?
22. Écrivez un fichier `test` qui contient un texte de votre choix. Compilez en orientant la sortie vers le fichier `test`. Que devient votre fichier ? Que se passera-t-il si par mégarde, vous compilez avec le fichier `.c` en sortie ? Testez le sur une copie de votre programme.

# Forcer l'arrêt d'un programme

23. Voici un programme qui ne va pas s'arrêter. Copiez le et compilez le pour tester la façon de forcer un arrêt :

```
#include <stdio.h>
int main(){
    long int compteur = 1;
    while(1){
        printf("%ld\n", compteur);
        compteur = compteur + 1;
    }
    return 0;
}
```

24. Compilez le programme. Lancez-le. Il va partir dans une boucle infinie avec de nombreux affichages de nombres. Pour interrompre le programme, vous pouvez utilisez le raccourci clavier CTRL+C. Testez.

# Affichage formaté

La fonction `printf` de la bibliothèque standard `<stdio.h>` vous permet de faire des affichages dans la console. On peut faire des affichages formatés avec des balises dans le texte. En particulier, la balise `%d` permet d'afficher un entier, `%f` un nombre à virgule [3].

2. indice : -o est un raccourci de `--output`
3. Souvent, on se met à parler de nombres *flottant* pour parler des nombres décimaux en machine, car il sont implémentés avec une virgule flottante.

25. Testez les affichages formatés dans la console en modifiant votre programme main pour ajouter les lignes suivantes :

```
int entier = 42;
double decimal = 5.1;
printf("Par exemple, %d est entier et %f est decimal.\n", entier,
  ↪ decimal);
```

26. Il existe un certain nombre de fonctionnalité dans l'usage de `printf` que vous pourrez être amenez à utiliser par la suite [4]. Testez (Prenez des notes, vous en aurez besoin plus tard !) :
    — l'affichage de nombre entiers avec une balise `%3d`, ou une variation de cette balise avec un autre nombre. À quoi sert cette balise ?
    — Une autre formulation qui ressemble, à savoir la notation `%03d`. Remarquez la différence de comportement.
    — l'affichage de nombre décimaux/flottants avec `%3f`, `%.3f`, `%5.2f`, `%05.2f`. Faites varier les nombres et identifiez le rôle de chacun.

# Renforcements

27. Vous allez utiliser un outil de suivi de l'activité processeur pour observer l'effet de ce programme. Ouvrez un autre terminal et lancez la commande `top`. Celle-ci vous permet de voir l'activité du processeur. Lancez le programme précédent dans le premier terminal et observez. Identifiez le numéro (PID pour processus ID) de votre programme.
28. Dans l'outil `top`, il est possible d'interrompre l'exécution du programme. Utilisez la touche 'k' (pour kill). Entrez le PID de votre processus pour le désigner, puis le code 15 pour le signal SIGTERM. Cela devrait interrompre le programme sans fermer l'autre terminal. Si l'autre terminal est fermé, réouvrez un terminal, relancez le programme infini, et observez mieux.
29. Si vous observez bien, vous pouvez constater que le total des usages de CPU dépasse 100%. En fait, chaque CPU apporte 100% d'usage possible. Vous pouvez lancer plusieurs instances du programme et observer le comportement... Vous devriez même pouvoir ainsi compter le nombre de CPU de votre machine.

Quand vous avez fini, continuez de vous entrainer à l'usage de la console avec le jeu *gameshell*. Suivez les instructions à cette adresse.

4. Au point que la fonction dispose même de sa propre page https ://fr.wikipedia.org/wiki/Printf.