1 Définitions

Avant de commencer, voyons quelques définitions.

Linux
Lorsque l'on dit **Linux**, on fait référence en fait au système d'exploitation (OS) **GNU/Linux**. *Linux* est en fait le noyau du système d'exploitation (*Kernel*). *Linux* (Kernel) n'offre aucun logiciel pour l'utilisateur (**userland**) par lui même. Le *kernel* offre seulement ce qu'on appelle des **system call** (**syscall**), qui sont des portes d'entrées vers des services offerts par le *kernel*. Pour raison de brièveté, lorsqu'il est écrit *Linux* dans ce documment, et ce à partir de maintenant, on fait référence à *GNU/Linux*. Lorsque l'on voudra faire référence au *kernel*, on en fera explicitement la remarque.

GNU
GNU est un accronyme récursif qui veut dire **GNU is Not Unix**. *GNU* est en fait un long projet (35 ans le 27 septembre) qui a pour but principal de faire un OS libre d'utilisation visant à remplacer **UNIX**. Je ne vais pas tarder sur le sujet, mais je vous invite fortement à en lire plus `https://www.gnu.org/gnu/thegnuproject.en.html`.

GNU/Linux
GNU/Linux est ce que vous utilisez au laboratoire. Vous utilisez le *kernel Linux* et vous utiliez comme *userland GNU*.

Distribution
Il est existe des centaines de distribution pour *Linux*. Une distribution est en fait un OS, basé sur *GNU/Linux* qui a une collection de *software* pré-installé. Une distribution a aussi ce que l'on appelle un **package manager**, qui s'assure, entre autre, de règler les conflits de dépendances entre les logiciels, installation/désintallation et des mises à jours de différents logiciels ou même le *kernel*. Chaque distribution a sa propre philosophie et son propre modèle de management des *packages*. Voici une liste non-exhaustive de distribution et une courte description pour chaque distribution.

2 Distributions

Ubuntu
Distribution probablement la plus poppulaire. Elle est très facile d'utilisation. Basée sur la distributon **Debian**, elle utilise donc le *package manager* **dpkg**. Utilise **GNOME3** ou **Ubuntu Unity** comme interface graphique par défaut. **Attention**, avant la version **16.04**, *Ubuntu* espionnait ses utilisateurs en vendant des informations à des troisièmes parties (comme Windows fait avec Cortana...). Ceci est en passant, la définition même d'un *spyware*. Vous pouvez lire à ce sujet: `https://www.gnu.org/philosophy/ubuntu-spyware.en.html`. *Ubuntu* est développé par la compagnie **Canonical**.

ElementaryOS
Distribution concentrée sur l'expérience utilisateur. Elle est basée sur *Ubuntu* et donc utilise *dpkg* comme *package manager*. L'interface graphique par défaut est **Pantheon** qui est sensiblement pareille que **Mac OS X**. Elle est donc très facile d'utilisation et est très jolie.

Solus
Solus est une nouvelle distribution (seulement 2 ans!) basée sur aucune autre distribution. Elle utilise comme *package manager* **eopkg**. Ce qui est intéressant avec *Solus*, c'est qu'il s'agit d'une **Rolling** distribution. Sans rentrer dans les détails, avec une distribution qui suit le modèle *rolling*, vous aurez toujours les dernières versions de vos logiciels, **driver** (*e.g* carte graphique), *kernel*, etc. Cela est aussi à double tranchant, vous aurez parfois des bugs dans vos logiciels car vous êtes en quelque sorte les premiers à tester la nouvelle mise à jour, mais cela est très rare et très vite corrigée. *Unbuntu* et *ElementaryOS* ne sont pas des *rolling* distribution, alors vous devez attendre quelque temps avant d'avoir de nouvelles mises à jour. Enfin, *Solus* a par défaut comme interface graphique soit, **Budgie**, **GNOME3** ou **MATE**. *Budgie* est fait maison par les développeurs de *Solus* et est très bon (quoi que j'ai eu quelques problèmes avec...).

Fedora
C'est la distribution qu'on a dans nos laboratoires à l'école. Je ne l'ai jamais utilisée alors je vais simplement dire qu'elle est développée par le projet *Fedora* qui est sponsarisé par la compagnie **Red Hat**.

Arch

Arch est une distribution pour les utilisatrices avancées seulement. Cette distribution est pour les processeurs à architecture **x86-64** seulement (des ports non officiels pour **i686** et **ARM** existent). Elle utilise comme *package manager* **pacman** et est une *rolling* distribution. Elle n'utilise aucun interface graphique par défaut (mode console). La philosophie de *Arch* est d'installer le strict minimum. Contrairement à d'autre distro qui installe pleins de truc innutile comme *Windows* (*Ubuntu* installe une application de Amazon par défault...), *Arch* installe le strict minimum (kernel + quelques autres trucs) et rien d'autre. Vous devez donc vous même, configurer votre *kernel* pour activer des services que vous voulez (internet en autre...), installer un environnement graphique, etc. L'avantage? Vous avez le contrôle totale sur votre distribution. *Arch* possède sont propre *Wiki* qui documente très bien plusieurs sujets.

3 File System (fs)

Sous *Linux*, les fichiers sont structurés dans un arbre. La racine de l'arbre est /. Pour accéder par exemple à mon dossier document, le chemin d'accès absolu est `/home/olivier/Documents`. Il n'y à pas de **C:** ou **D:** pour différencier les disques durs physiques comme sous *Window*. À la place, le **fs** va faire des points de montage virtuels à partir de la racine. Par exemple, si je veux monter un disque dur contenant ma musique et des films, je vais créer un dossier nommé `~/Media` et je vais monter le disque à ce point. Ici, `~` signifie mon répertoire maison (`/home/olivier` sur mon ordie). Voici un exemple complet.

```
$ mkdir ~/Media
$ sudo mount /dev/sda ~/Media
```

Ici, on a créé le dossier Media dans le dossier `~` et on a monté le disque dur A dans le dossier `~/Media`.

4 Process

À COMPLETER

5 Variable d'environnement

Chaque *process* possède une liste de variable d'environnement. Il s'agit d'une liste de pair **key:value**. Un *process* peut modifier à tout moment sa liste. Lorsqu'un *process* est créé, celui-ci copie la liste d'environnement de son parent.

```
# Pour voir la liste d'environnement d'un shell
$ printenv
```

6 Shell & Terminal

6.1 Terminal

Sans trop rentrer dans les détails, lorsque vous ouvrez un terminal sous *Linux*, il s'agit d'un programme qui émule un vrai terminal. Le terminal effectue une communication entre le programme qui le contrôle et un **device**. Par exemple, mon terminal à la maison est connecté au *device* `/dev/pts/0` qui est un pseudo terminal. Le vrai terminal de la session c'est le serveur **XWindow** (interface graphique) qui est connecté à `/dev/tty7`. C'est correct si vous ne comprenez pas trop, l'important est de se rappeler qu'un terminal est seulement là pour contrôler les inputs/outputs. Il ne regarde pas si vous avez entrer la commande **ls** par exemple.

6.2 Shell

Un *Shell* est un programme qui se connecte à un terminal ou pseudo terminal (faisant de lui un **session leader**) et qui interprète des commandes que vous envoyez à travers le terminal. Par défaut quand vous ouvrez un terminal, celui-ci va executer le programme se trouvant au chemin d'accès définit dans la variable d'environnement **SHELL**. Vous pouvez changer ce comportement en configurant votre terminal. Par exemple chez moi, mon terminal ouvre par défaut le programme au chemin d'accèes **/bin/bash**. *Bash* est pour **Bourne Again Shell** et est un *shell* développé par *GNU*. Il existe d'autre *shell* tels que **sh**, **dash**, **zsh**. Chaque *shell* possède un language de programmation qui se ressemble, mais peut différer sur certains points.

7 Bash

Par défaut, *Bash* se place dans le chemin d'accès contenue dans la variable d'environnement **HOME**. Si cette variable n'existe pas ou bien que le chemin d'accès est inexistant ou que vous n'avez pas les droits d'accès, *Bash* va soit créer le dossier pour vous, ou vous placer dans `~`. *Bash* afficher votre nom d'utilisateur suivit du nom de l'*host*, le chemin d'accès dans lequel vous vous trouvez et dans quel mode vous vous trouvez. Voici une capture d'écran pour vous montrer.

olivier est mon nom d'utilisateur, **clara** est le nom d'*host* de mon ordinateur. Je me trouve dans le répertoire `/home/olivier` et je suis en mode utilisateur normal **$**. Si je veux aller en mode administrateur (**root**), je peux faire la commande **sudo su**. Ne le faite pas sur les poste à l'école!

Comme vous pouvez le voir, sur *Linux* la sécurité est très importante. Et donc, lorsque l'on demande mon mot de passe, on n'affiche rien, même pas de *****. Comme ça, vous ne pouvez pas savoir la longueur de mon mot de passe! De plus, le **$** a été remplacé par # car je suis maintenant connecté en tant que **root**.

7.1 Lister les fichiers

Avant de nous déplacer, commençon par lister le dossier dans lequel on se trouve.

```
olivier@clara /home/olivier $ ls -l
total 92K
drwxr-xr-x  2 olivier olivier 4.0K May  8 02:31  Desktop/
drwxr-xr-x 18 olivier olivier 4.0K Sep  9 14:10  Documents/
drwxr-xr-x  8 olivier olivier 4.0K Aug 14 19:43  dotfiles/
drwxr-xr-x  3 olivier olivier 4.0K Sep  9 14:05  Downloads/
drwx------  9 olivier olivier 4.0K Aug 13 14:29  emacs/
drwxrwxr-x  3 olivier olivier 4.0K Sep  9 16:18  foo/
drwxrwxr-x  2 olivier olivier 4.0K Aug 14 00:58  Iso/
drwxr-xr-x  8 olivier olivier 4.0K Jul 28 15:26  Music/
drwxr-xr-x  3 olivier olivier 4.0K Sep  1 21:30  Pictures/
drwxr-xr-x  2 olivier olivier 4.0K May  7 23:00  Public/
drwxrwxr-x  4 olivier olivier 4.0K Jul 29 15:37  Sandbox/
drwxrwxr-x  2 olivier olivier 4.0K Jul  5 14:34  SteamLibrary/
drwxrwxr-x  2 olivier olivier 4.0K Aug 14 17:24  Tar/
drwxr-xr-x  2 olivier olivier 4.0K May  7 23:00  Templates/
drwxr-xr-x  2 olivier olivier 4.0K May  7 23:00  Videos/
drwxr-xr-x  2 olivier olivier 4.0K Aug 14 01:17  VirtualBoxShared/
drwxr-xr-x  3 olivier olivier 4.0K Aug 14 00:43 'VirtualBox VMs'/
-rw-rw-r--  1 olivier olivier   18 Jun  5 20:12  banq
-rw-rw-r--  1 olivier olivier  237 Sep  4 19:02  books
-rwxr-xr-x  1 olivier olivier   64 Aug 14 20:02  ping.bash
-rw-rw-r--  1 olivier olivier  504 Sep  4 22:41  timing
-rw-rw-r--  1 olivier olivier 2.0K Sep  4 22:41  typescript
-rwxrwxr-x  1 olivier olivier  109 Sep  4 20:10  vpn-poly
olivier@clara /home/olivier $
```

La dernière colonne est le nom des fichiers dans le dossier courrant. Ici, j'ai customisé mon *shell* pour lister les dossiers en bleu, les fichiers ordinaires en blanc et les exécutables en vert. La première colonne représente les permissions d'accès au fichier. **d** veut dire *directory*. **r** veut dire *read*, **w** write et **x** *execute*. Un **-** veut dire aucune permission. Les permissions sont répéter 3 fois. La première fois pour l'utilisateur a qui appartient le fichier, la deuxième fois pour le groupe auquel le fichier appartient et enfin le dernier pour les autres utilisateurs. La deuxième colonne représente le nombre de lien vers le fichier. La troisième et quatrième colonne représentent respectivement l'utilisateur et le groupe propriétaire du fichier. La cinquième colonne donne la taille du fichier en octet. Enfin, de la sixième à la huitième colonne, il s'agit du *timestamp* du fichier.

Par exemple, le fichier **banq** est un fichier ordinaire (**-**). L'utilisateur et le groupe propriétaire ont le droit de lire et d'éditer le fichier **rw-** et les autres utilisateurs ont seulement le droit de le lire **r–**. Le propriétaire du fichier est **olivier** et le groupe propriétaire est **olivier**. Le fichier pèse 18 octets et a été modifié la dernière fois le 5 Juin à 20:12.

Les fichiers cachés non pas été lister par contre. Un fichier caché commence par un point (**.**). Pour lister les fichiers cachés, il faut inclure dans la commande **ls** l'option **-a** (**all**).

```
olivier@clara /home/olivier $ ls -a
total 324K
drwxr-xr-x 44 olivier olivier 4.0K Sep  9 17:14  ./
drwxr-xr-x  4 root    root    4.0K May 25 07:03  ../
drwx------  3 olivier olivier 4.0K Sep  3 16:24  .adobe/
drwxr-xr-x  3 olivier olivier 4.0K Aug 13 23:57  .antidote/
drwx------ 42 olivier olivier 4.0K Sep  9 15:54  .cache/
drwxr-xr-x 43 olivier olivier 4.0K Sep  9 16:32  .config/
drwx------  3 olivier olivier 4.0K May  7 23:00  .dbus/
drwxr-xr-x  2 olivier olivier 4.0K May  8 02:31  Desktop/
drwxr-xr-x  5 olivier olivier 4.0K Aug  9 18:42  .dia/
drwxr-xr-x 18 olivier olivier 4.0K Sep  9 14:10  Documents/
drwxr-xr-x  8 olivier olivier 4.0K Aug 14 19:43  dotfiles/
drwxr-xr-x  3 olivier olivier 4.0K Sep  9 14:05  Downloads/
drwxr-xr-x  2 olivier olivier 4.0K Aug 13 21:50  .Druide/
drwx------  9 olivier olivier 4.0K Aug 13 14:29  emacs/
drwx------  9 olivier olivier 4.0K Sep  9 14:13  .emacs.d/
drwxrwxr-x  3 olivier olivier 4.0K Sep  9 16:18  foo/
drwx------  3 olivier olivier 4.0K Aug  9 18:41  .gnome2/
drwx------  2 olivier olivier 4.0K Aug  9 18:41  .gnome2_private/
drwx------  3 olivier olivier 4.0K May 28 16:06  .gnupg/
drwxr-xr-x  2 olivier olivier 4.0K Aug 14 17:57  .i3/
drwxrwxr-x  2 olivier olivier 4.0K Aug 14 00:58  Iso/
```

On voit qu'il y avait beaucoup de fichiers cachés (la liste continue encore en bas). Vous remarquerez aussi les deux fichiers spéciaux **.** et **..**. **.** signifie le répertoire courrant tandit que **..** est le répertoire parent. Donc, dans `/home/olivier`, **..** représente `/home` et **.** `/home/olivier`. Enfin, pour afficher tous les fichiers sauf **.** et **..**, utiliser l'option **-A** (**Almost all**).

7.1.1 Chemin d'accès absolu et relafif

Un chemin d'accès absolu commence toujours par la racine. Un chemin d'accès relatif commence soit par `.` ou bien `..`. Par exemple,
`/usr/include/SDL2/SDL_vulkan.h` est un chemin absolu tandit que
`../Documents/Poly/LOG1000/README` et
`./patrick_watson/close_to_paradise/the_great_escape.mp3` sont des chemins relatifs. Dans certain cas, *Bash* va interpréter un chemin ne remplissant aucune des conditions précédentes comme un chemin relatif au dossier courrant. Par exemple,
`Dossiers/Poly` va être vue comme `~/Dossiers/Poly` si je suis dans `~`.

8 Se Promener

La commande **cd** (**Change Directory**) permet de changer de répertoire courrant. On peut soit passer un chemin absolu ou bien un chemin relatif. Si aucun chemin n'est spécifié, *Bash* retourne à **HOME**. On peut aussi retourner au dernier dossier visité à l'aide de **cd -**.

```
olivier@clara ~ $ cd Documents/linux-4.17-rc7/
olivier@clara ~/Documents/linux-4.17-rc7 $
```

9 Supprimer

Pour supprimer des fichiers, on utilise la commande **rm** (**remove**). **Attention**, cela ne place pas les fichiers dans la corbeille. Ceux-ci sont perdus à jamais (ou presque). Pour supprimer un dossier (et donc les fichiers dans le dossier), on doit passer l'option **-r** (**recursive**) à la commande.

```
olivier@clara ~/bar $ ls -R .
.:
total 4.0K
drwxrwxr-x 2 olivier olivier 4.0K Sep  9 17:46 foo/

./foo:
total 0
-rw-rw-r-- 1 olivier olivier 0 Sep  9 17:46 main.c
-rw-rw-r-- 1 olivier olivier 0 Sep  9 17:46 main.h
-rw-rw-r-- 1 olivier olivier 0 Sep  9 17:46 makefile
olivier@clara ~/bar $ rm -r foo/
olivier@clara ~/bar $ ls -R .
.:
total 0
olivier@clara ~/bar $
```

10 Grep

La commande **grep** est un outil extrêmement puissant qui vous permet de chercher une expression dans des fichiers. **grep** (la version de *GNU*) accepte aussi les expressions régulières.

```
olivier@clara ~/Documents/Poly/INF1010/TP1/inc $ ls -l
total 16K
-rw-rw-r-- 1 olivier olivier  560 Sep  8 11:58 depense.h
-rw-rw-r-- 1 olivier olivier 1.2K Sep  8 11:58 groupe.h
-rw-rw-r-- 1 olivier olivier  662 Sep  8 11:58 transfert.h
-rw-rw-r-- 1 olivier olivier  865 Sep  8 11:58 utilisateur.h
olivier@clara ~/Documents/Poly/INF1010/TP1/inc $ grep class *.h
depense.h:class Depense {
groupe.h:class Groupe{
transfert.h:class Transfert {
utilisateur.h:class Utilisateur {
olivier@clara ~/Documents/Poly/INF1010/TP1/inc $
```

11 Sed

À COMPLETER

12 Awk

À COMPLETER

13 GCC

À COMPLETER

14 Emacs

À COMPLETER

15 Manuel

La commande **man** vous permet de lire le manuel d'une commande, d'un programme, d'un syscall, etc. **man** est divisé en 7 sections.

1 Commandes générales

2 Syscall

3 Fonctions de libraries

4 Fichiers spéciaux

5 Formats de fichiers

6 Jeux

7 Autres

8 Administration du système

Par exemple, si vous voulez lire sur **write**. Il existe **write(1)**, mais aussi **write(2)**. Par défaut, **man** va ouvrir la première page du manuel quelle trouve (**write(1)**). Si vous voulez lire **write(2)**, vous devez spécifier 2 comme argument.

```
$ man write   # Va ouvrir write(1)
...
$ man 2 write # Va ouvrir write(2)
```

Vous pouvez ainsi lire le manuel des commandes parlées précédament, par exemples **man ls** ou bien **man grep**. Vous pouvez aussi lire le manuel de **man**. **man man**.

16 Help

Si vous êtes perdu, vous pouvez tapper la commande **help** dans le terminal. Une liste de commande sera alors affichée. Utilisé ceci avec **man** et vous devriez être correct.