- Introduction à l'algorithmique et au langage C -

Initiation au Shell

TD1

1re année ESIEA - Semestre 1

L. Beaudoin & R. Erra & A. Gademer & L. Avanthey

2016 - 2017

Avant propos

Ce TD aura pour objet de présenter très rapidement la partie ***software*** *de l'ordinateur. Nous aborderons entre autres l'organisation des fichiers et la gestion des droits utilisateurs. Vous découvrirez aussi un outil extrêmement puissant : le* `Terminal`*, qui permet d'exécuter des commandes* `Shell` *et ainsi d'interagir avec l'ensemble de votre système.*

1 Organisation des informations dans le système

1.1 Arborescence de fichiers

Tous les systèmes d'exploitation reposent sur un principe identique : les informations sont organisées sous forme de *fichiers* (files) que nous hiérarchisons par des *dossiers* (directories), aussi appelés répertoires. Un dossier peut rassembler des fichiers ou d'autres dossiers, ce qui conduit à obtenir une *arborescence*.

— Le dossier dans lequel nous nous trouvons et dont nous pouvons voir le contenu est appelé **dossier courant**.
— Les dossiers qu'il contient sont ses **dossiers fils**.
— Le dossier qui le contient s'appelle le **dossier parent**.
— Le dossier qui contient tous les autres dossiers s'appelle **la racine**.

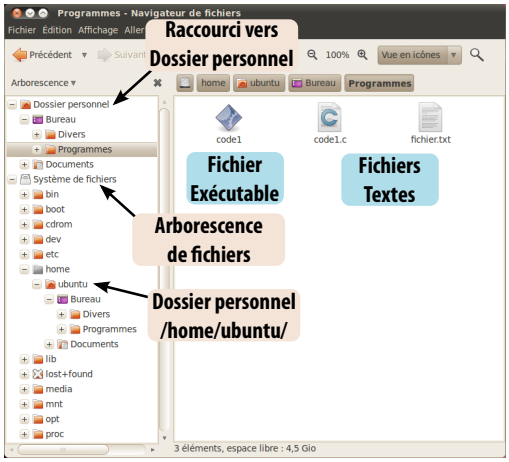

Figure 1 – ***Arborescence vue dans le navigateur Nautilus.***

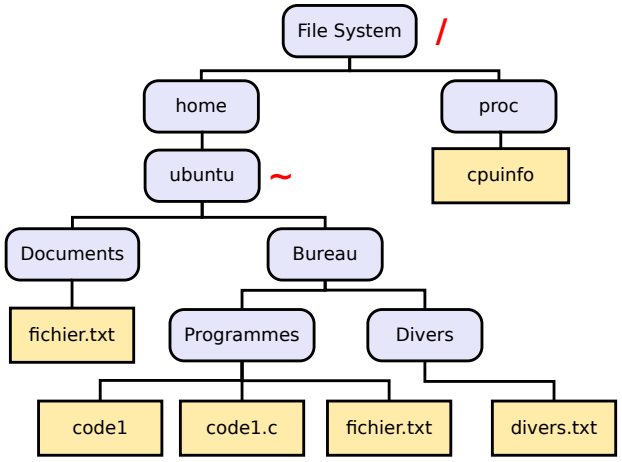

Figure 2 – ***Arborescence représentative.***

QUESTION 1


Associez ces différents mots-clés à l'arborescence donnée en exemple dans la figure 2.

Mot-clé	Nom du dossier concerné
Dossier courant	`Bureau`
Dossier parent	
Dossiers fils	
Racine	

Saviez-vous ?

Sous `Microsoft Windows`, il y a **plusieurs répertoires racines**. Chaque disque dur possède sa propre racine : `C:`, `E:`, etc.. Dans les systèmes UNIX, en revanche, le système possède une **racine unique**, et les disques durs n'apparaissent que secondairement dans le sous-dossier `mnt` (« *mount* »).

1.2 Fichier où es-tu ?

Dans notre arborescence d'exemple, nous avons deux fichiers qui portent le même nom. Si nous parlons de « `fichier.txt` », nous ne pouvons pas savoir auquel nous faisons référence !

Mais un élément les différencie : ils ne sont pas dans le même répertoire. Nous pouvons donc utiliser cette information pour les distinguer. Mais ces répertoires peuvent eux-même posséder des homonymes. Il faut à nouveau les distinguer par leurs répertoires parents, et ainsi de suite jusqu'à la racine.

Cette liste de dossiers s'appelle un **chemin** (path). Nous séparons chacun des dossiers de la liste par des barres obliques (slash). Ainsi la racine qui ne possède pas de répertoire parent sera désignée par une barre oblique unique : « / ».

Par exemple, dans l'arborescence des figures 1 et 2 le chemin du fichier qui se trouve dans le dossier « Documents » serait :

```
/home/ubuntu/Documents/fichier.txt
```

QUESTION 2


Écrivez le chemin de l'autre fichier `fichier.txt` qui se trouve dans le dossier « `Programmes` ».

Attention

Sous les systèmes `UNIX` le séparateur entre les dossiers est la barre oblique (slash) `/` (comme pour les `URL`) alors que sous `Microsoft Windows` c'est la barre oblique inversée (antislash) `\` qui tient ce rôle.

Nous ne sommes pas toujours obligés de lister tous les dossiers depuis la racine, nous pouvons aussi créer cette liste à partir du répertoire courant (ce qui dans certains cas nous permettra d'écrire des chemins plus courts). Nous utilisons pour cela une référence spéciale que vous retrouverez dans la table 2.

Table 2 – ***Références spéciales.***

./	Le répertoire courant
../	Le répertoire parent
~/	Le dossier personnel de l'utilisateur

QUESTION 3

 En considérant que notre répertoire courant est le répertoire « Dossier personnel », écrivez les chemins de chacun des deux fichiers « `fichier.txt` » (utilisez la référence spéciale pour « répertoire courant »).

QUESTION 4

 Même question, mais le répertoire courant est « `Bureau` ».

QUESTION 5

 Même question, mais le répertoire courant est maintenant « `Documents` ».

QUESTION 6

 Donnez maintenant le chemin des deux mêmes fichiers mais à partir du dossier personnel de l'utilisateur (`~/`) qui est ici « ubuntu » en utilisant la référence spéciale.

QUESTION 7

 Simplifiez le chemin `~/Bureau/Programmes/../../Documents/fichier.txt` ?

Chemin absolu, chemin relatif

Nous appellerons tous les chemins qui partent de la racine des **références absolues**. Elles sont **uniques**.

Les autres chemins sont des **références relatives** et dépendent du répertoire de référence (dossier courant ou dossier personnel de l'utilisateur).

2 Jouer à la console

2.1 Premiers pas

Maintenant que nous savons décrire des chemins, nous allons pouvoir les utiliser pour effectuer des actions avec des commandes. Nous avons prétendu un peu plus tôt que tout ce que nous permet de réaliser le serveur graphique, nous pouvions le faire en lignes de commande Shell. Nous allons commencer par prendre en main l'outils `Terminal`, puis nous verrons comment nous pouvons visualiser l'arborescence, nous y déplacer et la modifier (ajout de dossiers ou de fichiers, déplacement, etc.).

EXERCICE 1

Lancez un `Terminal` (`Menu Applications > Accessoires > Terminal`).

Figure 3 – ***Le Terminal permet d'accéder aux commandes Shell.***

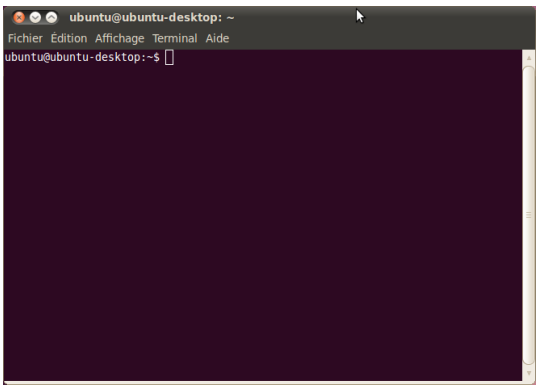

Figure 4 – ***Une fois ouvert, le Terminal attend vos instructions.***

Où suis-je ?

Quand vous ouvrez un nouveau terminal, il choisit toujours comme dossier courant le dossier personnel de l'utilisateur (`~/`). Il vous le marque d'ailleurs juste avant le caractère « `$` » à la fin de la ligne. Le voyez-vous ?

Le `Terminal`, que nous appelons aussi `Console` est un programme qui vous permet de lancer des commandes `Shell` et donc d'interagir avec votre système.

Trucs & Astuces

Vous pouvez vous faire un raccourci pour votre terminal avec :
`Clic droit > Ajouter au tableau de bord`).

Comme vous pouvez le voir sur la figure 4, une fois ouvert, le `Terminal` vous propose une zone de texte et semble attendre vos instructions. Nous appelons cette configuration **l'invite de commande** (prompt en anglais).

En effet, contrairement au serveur graphique qui utilise principalement les boutons de la souris pour interagir avec les programmes, le `Terminal` ne fonctionne que par l'échange de commandes textuelles.

EXERCICE 2

Tapez pour commencer la commande `date` suivie de la touche `Entrée`.

```
ubuntu@ubuntu-desktop:~$ date
lundi 5 septembre 2011, 08:30:00 (UTC+0200)
```

Que s'est-il passé ? Vous avez demandé au Terminal d'exécuter le programme `date`. Ce dernier a renvoyé une chaîne de caractères décrivant la date actuelle au Terminal qui l'a affichée.

Premier super-pouvoir du Terminal

Il peut être fastidieux de taper de longues lignes de commande sans se tromper ! En informatique, nous adorons l'efficacité et la rapidité, alors votre `Terminal` possède la capacité d'**auto-complétion** !

En appuyant une fois sur la touche `tabulation`, il complétera autant que possible la commande que vous avez commencé à taper.

```
ubuntu@ubuntu-desktop:~$ dat(tab)
ubuntu@ubuntu-desktop:~$ date
```

En appuyant deux fois sur `tabulation`, il vous proposera toutes les possibilités que vous avez.

```
ubuntu@ubuntu-desktop:~$ da(tab)(tab)
dash date
ubuntu@ubuntu-desktop:~$ da
```

QUESTION 8

Combien y a-t-il de commandes qui commencent par la lettre `g` ?

QUESTION 9

Combien y a-t-il de commandes au total ?

Deuxième super-pouvoir du Terminal

Toujours pour les mêmes raisons, nous détestons réécrire les mêmes commandes à chaque fois que nous en avons besoin. Rassurez-vous : en appuyant sur la touche `HAUT`, vous accéderez aux dernières commandes que vous avez tapées ! Et si la commande est un peu loin dans la liste, utilisez `CTRL+R` (suivi du début de la commande) pour chercher directement dans l'historique.

EXERCICE 3

Essayez CTRl+R puis tapez « da », la commande date vous sera proposée :

```
(reverse-i-search)`da': date
```

Voici quelques commandes Shell. Celles marquées d'une étoile * sont **à retenir absolument**.

TABLE 3: Commandes Shell générales

Nom	Commande	Effet
DATE	`date`	Affiche la date courante.
	`date +"%Hh%M"`	Affiche uniquement l'heure et les minutes.
	`date +%A -d "2011/12/24"`	Affiche le jour de la semaine correspondant à la date fournie (ici, noël prochain).

(Suite page suivante)

Nom	Commande	Effet
MAN	`man CMD*`	Affiche l'aide sur la commande CMD.
	`man -a CMD`	Affiche **toutes** les pages d'aide sur CMD.
	`man -k XXX`	Recherche XXX dans **toutes** les pages d'aide.
WHICH	`which CMD`	Affiche le chemin vers le programme associé à CMD.
UNAME	`uname -a`	Affiche les caractéristiques du système d'exploitation (Version du noyau, type de processeur...).

EXERCICE 4

Essayez tout de suite la commande `man date`. Le résultat de la commande est illustré à la figure 5.

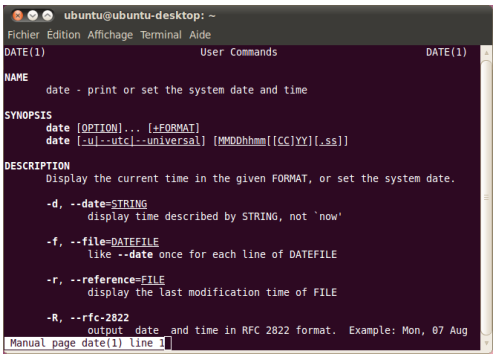

Figure 5 – *La commande man permet d'afficher l'aide sur les commandes Shell*

Lorsque vous êtes dans le manuel, vous pouvez :
- vous déplacer en utilisant les flèches,
- quitter en appuyant sur la touche `Q`
- rechercher une information en tapant la touche `slash` suivi du mot recherché puis `Entrée`. par ex : `/%Y(Entrée)` pour chercher le sens du descripteur `%Y`.
- aller à l'occurrence suivante du mot recherché avec la touche `N`.

QUESTION 10

À quoi sert le descripteur `%N` de la commande `date` ?

Saviez-vous ?

La plupart des utilitaires du Terminal fournissent de l'aide lorsque nous leur passons l'argument `-h` ou `--help`. Essayez par exemple : `date --help`.

2.2 Naviguons dans l'arborescence et manipulons-la

2.2.1 Voir et se déplacer

La première chose que nous devons apprendre, c'est à nous déplacer dans l'arborescence. Voici à cet effet une liste de commandes `Shell` qui vous seront très utiles. Les commandes marquées d'une étoile * sont **à retenir absolument**.

TABLE 4: Commandes `Shell` liées à l'arborescence : listes et déplacements.

Nom	Commande	Effet
Path Working Directory*	`pwd`	Affiche le chemin absolu jusqu'au répertoire courant.
List Segment	`ls`*	Affiche le contenu du répertoire courant.
	`ls DIR_PATH`*	Affiche les fichiers contenus dans le dossier spécifié par le chemin.
	`ls PATTERN`*	Affiche les fichiers qui correspondent au motif PATTERN.
	`ls -a`	Affiche les fichiers cachés.
Change Directory*	`cd DIR_PATH`	Saute jusqu'au dossier spécifié par le chemin donné.
conCATenate	`cat FILE`*	Affiche le contenu du fichier FILE
	`cat FILE1 FILE2`	Affiche le contenu du fichier FILE1 immédiatement suivi du contenu de FILE2
FILE	`file FILE`	Affiche le type du fichier FILE.

EXERCICE 5

 Ouvrez un terminal et tapez la commande qui vous permet de connaître le chemin absolu du dossier dans lequel vous vous trouvez.

EXERCICE 6

 Tapez maintenant la commande qui vous permet d'afficher le contenu du dossier dans lequel vous vous trouvez.

EXERCICE 7

 Tapez la commande qui vous permet de vous déplacer dans le dossier Bureau. Affichez son contenu, puis revenez au dossier de départ.

EXERCICE 8

 Affichez le contenu du dossier Bureau depuis le dossier courant (~/), sans vous y déplacer (vous devez donc indiquer un chemin).

EXERCICE 9

 Comment afficher le contenu du fichier `cpuinfo` qui est dans le dossier `/proc` ?

Troisième super-pouvoir du Terminal

Pour toutes les commandes qui attendent des fichiers ou des dossiers, plutôt que de leur indiquer un nom exact et unique, vous pouvez leur spécifier des motifs (pattern en anglais). Vous allez adorer le Joker `*` qui signifie « n'importe quelle suite de caractères ».

Ainsi, nous pouvons afficher la liste des fichiers dont les noms se terminent par « .jpg » :

```
ls *.jpg
```

Ou bien tous les fichiers qui contiennent dans leur nom la chaîne de caractères « toto » :

```
ls *toto*
```

QUESTION 11

 Comment afficher la liste de tous les fichiers contenus dans `/usr/include/` et commençant par `std` ?

2.2.2 Créer et supprimer

TABLE 5: Commandes `Shell` liées à l'arborescence : création & suppression.

Nom	Commande	Effet
MaKe DIRectory	`mkdir DIR`*	Crée le répertoire DIR.
	`mkdir -p DIR`	Crée le dossier DIR ainsi que tous les dossiers parents nécessaires (s'ils n'existent pas).
ReMove DIRectory*	`rmdir DIR`	Supprime le répertoire DIR (s'il est vide).
TOUCH	`touch FILE`	Met à jour la date de modification du fichier FILE et le crée s'il n'existe pas (nous l'utilisons couramment pour **créer un fichier vide**).
ReMove	`rm FILE`*	Supprime le fichier FILE.
	`rm PATTERN`	**DANGEREUX !** Supprime tous les fichiers correspondant au motif PATTERN.
	`rm -r DIR`*	**DANGEREUX !** Supprime le répertoire DIR ainsi que tout son contenu (fichiers et sous-répertoires).

EXERCICE 10

 À partir du terminal, rendez-vous dans le dossier `Bureau`. Créez-y un dossier « `Programmes` » et un dossier « `Divers` ». Vérifiez leur présence en affichant le contenu du dossier « `Bureau` ».

EXERCICE 11

 Dans le dossier « `Divers` », créez un fichier : « divers.txt ». Vérifiez sa présence en affichant le contenu du dossier « `Divers` ».

EXERCICE 12

 Dans le dossier « `Programmes` », créez trois fichiers :« `fichier.txt` », « `code1.c` » et « `code2.c` ». Vérifiez leur présence en affichant le contenu du dossier « `Programmes` ».

Mauvaise pratique

L'espace n'est pas la bienvenue dans les chemins, car elle permet avant tout de séparer les arguments (FILE1 et FILE2 par exemple).

QUESTION 12

 Pour illustrer cette remarque, créez le dossier « `Espace de travail` » dans le dossier « `Bureau` ». Affichez le contenu du dossier « `Bureau` ». Que s'est-il passé ? Supprimez ce que vous venez de créer.

2.2.3 Déplacer, copier et renommer

TABLE 6: Commandes `Shell` liées à l'arborescence : copie, déplacement & raccourcis.

Nom	Commande	Effet
MoVe*	`mv FILE DIR`	Déplace le fichier FILE vers DIR
	`mv FILE1 FILE2`	Déplace et renomme le fichier FILE1 en FILE2.
CoPy*	`cp FILE DIR`	copie le fichier FILE vers DIR
	`cp FILE1 FILE2`	Copie et renomme le fichier FILE1 en FILE2.
LiNk	`ln -s FILE1 FILE2`	Crée un lien symbolique (un raccourci) appelé FILE2 et pointant sur FILE1.
	`ln -s DIR2 DIR2`	Crée un lien symbolique (un raccourci) appelé DIR2 et pointant sur DIR1.

EXERCICE 13

 Déplacez le fichier `code1.c` dans le dossier « Divers ». Vérifiez votre action en affichant le contenu des dossiers « Divers » et « Programmes ».

EXERCICE 14

 Renommez le fichier `code1.c` en `monCode.c`. Vérifiez votre action en affichant le contenu du dossier « Divers ».

EXERCICE 15

 Copiez le fichier `monCode.c` dans le dossier « Programmes ». Vérifiez votre action en affichant le contenu des dossiers « Divers » et « Programmes ».

EXERCICE 16

 Dupliquez fichier `monCode.c` en `monCodeBis.c`. Vérifiez votre action en affichant le contenu du dossier « Programmes ».

Trucs & Astuces
Pour « effacer » le contenu du terminal sans avoir à en ouvrir un nouveau, utilisez CTRL+L (ou tapez `clear`).

3 Qui peut faire quoi ?

Nous avons vu dans la section précédente qu'il était extrêmement simple de lire (`cat`) ou de supprimer (`rm`) des fichiers. Or les systèmes `UNIX` sont intrinsèquement des systèmes d'exploitation **multi-utilisateurs** et nous ne voulons certainement pas qu'un autre utilisateur lise ou détruise nos fichiers ! Dès lors, une question se pose.

Qui peut faire quoi ?

Pour un fichier ou un répertoire donné, nous pouvons donner des autorisations de lecture, d'écriture et d'exécution à trois groupes d'utilisateur : le propriétaire du fichier (User), son groupe (Group) et tous les autres (Other). Ce qui nous donne au total neuf permissions pour chaque ressource :

	User	Group	Other
Read	?	?	?
Write	?	?	?
eXecute	?	?	?

Voici les commandes liées à la gestion de permission. Celles marquées d'une étoile * sont **à retenir absolument**.

TABLE 7: Commandes `Shell` liées aux permissions.

Nom	Commande	Effet
List Segment	`ls -l`*	Affiche les fichiers sous forme de liste détaillée
	`ls -l -h`	Affiche les fichiers sous forme de liste détaillée et donne la taille en Ko ou Mo.
CHange MODe	`chmod PERM FILE`*	Change les permissions du fichier FILE.
	`chmod PERM DIR`*	Change les permissions du dossier DIR.
	`chmod -R PERM DIR`	Change les permissions du dossier DIR **et de tout son contenu**, les sous-dossiers et leur contenu inclus.
CHange OWNer	`chown USER:GROUP FILE`	Change le propriétaire et le groupe du fichier FILE.
	`chown USER:GROUP DIR`	Change le propriétaire et le groupe du dossier DIR.
	`chown -R USER:GROUP DIR`	Change le propriétaire et le groupe du dossier DIR **et de tout son contenu**, les sous-dossiers et leur contenu inclus.

EXERCICE 17

 Allez dans le dossier « `Programmes` » et lancez la commande `ls -l` pour afficher la liste détaillée et voir les permissions des fichiers qu'il contient.

Vous obtenez quelque chose qui ressemble à ça :

```
total 0
-rw-r--r-- 1 ubuntu ubuntu 0 29 sep 20:56 code2.c
-rw-r--r-- 1 ubuntu ubuntu 0 29 sep 20:55 fichier.txt
-rw-r--r-- 1 ubuntu ubuntu 0 29 sep 20:58 monCode.c
-rw-r--r-- 1 ubuntu ubuntu 0 29 sep 20:58 monCodeBis.c
```

Voici les informations que nous obtenons :
- **Premier caractère** : le *type de ressource* (`-` pour un fichier, `d` pour un dossier, `l` pour un lien symbolique, etc.)
- **Trois caractères suivants** : statuts des *trois permissions de l'utilisateur* `rwx` (la lettre est remplacée par `-` si la permission n'est pas accordée).
- **Trois caractères suivants** : statuts des *trois permissions du groupe* (même principe).
- **Trois caractères suivants** : statuts des *trois permissions de tous les autres utilisateurs* (même principe).

Après cela, vous trouvez les informations telles que le nom du propriétaire, le nom du groupe, la taille en octet et la date de la dernière modification suivie du nom du fichier ou du répertoire.

Comment marche la commande `chmod` ?

PERM correspond à une succession de blocs séparés par des virgules, dont voici le schéma : (cible)(opérateur)(permission)

- (cible) : a, u, g ou o (pour all, user, group ou other) ou une combinaison de plusieurs d'entre eux,
- (opérateur) : +, - ou = (pour ajouter, retirer ou fixer),
- (permission) : r, w, x (pour read, write, execute) ou une combinaison de plusieurs d'entre eux.

Par exemple, la commande suivante permet de donner tous les droits (lecture, écriture et exécution) à tout le monde, puis de retirer le droit d'exécution au groupe et aux autres :

```
chmod a+rwx,go-x monCode.c
```

EXERCICE 18

 Testez la commande. Lancez à nouveau la commande `ls -l` et notez les changements pour le fichier `monCode.c`.

Autre exemple :

```
chmod u=rwx,g=rw,o= monCodeBis.c
```

QUESTION 13

 Lancez cette commande. Qu'avons nous changé comme droits ?

QUESTION 14

 Est-il possible de faire la distinction si nous appartenons tous au même groupe ?

Saviez-vous ?

Le système de permission ne s'applique qu'au niveau des applications (user space) mais pas dans le noyau (kernel space) où tous les programmes peuvent accéder à toutes les ressources. C'est pourquoi ce dernier est l'objet de toutes les protections (et de toutes les convoitises).

4 Jouons au Lego®

N. Matthew et al., *Programmation Linux*

« UNIX est construit de façon à user et abuser du code existant. Si vous créez un petit utilitaire, même élémentaire, d'autres peuvent s'en servir dans une chaîne comme lien vers d'autres utilitaires pour former une commande. [...] Chaque utilitaire se comporte comme une brique : en réalité, les programmeurs UNIX n'ont jamais cessé de jouer au Lego® ! »

4.1 Flux d'informations

Nous avons vu que le `Terminal` fonctionnait comme un tchat entre nous et le système (plus précisément ses programmes). Nous tapons des commandes, le `Terminal` envoie ce qu'il faut à un utilitaire et lui demande de s'exécuter, puis il récupère l'information renvoyée par ce dernier et nous l'affiche dans la console.

Les informations envoyées au programme sont appelées **flux d'entrée** du programme (`stdin` pour STandarD INput).

Les informations renvoyées par le programme sont appelées **flux de sortie** du programme (`stdout` pour STandarD OUTput).

Il existe un troisième flux, appelé **flux d'erreur** (`stderr` pour STandarD ERRor) et qui reçoit les messages d'erreur renvoyés par le programme.

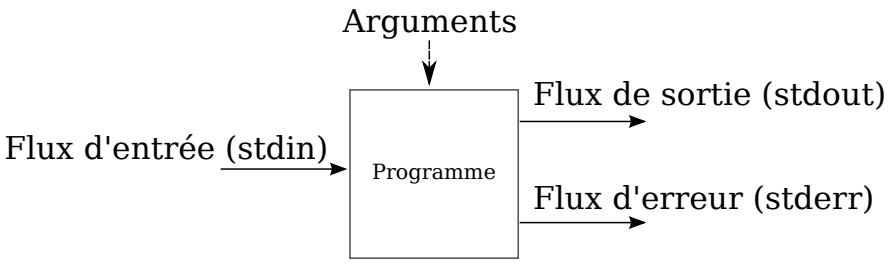

Figure 6 – ***Tous les programmes interagissent par l'intermédiaire des flux d'informations.***

Comportement par défaut

Par défaut, le flux d'entrée correspond à la saisie clavier, alors que les flux de sortie et d'erreurs sont affichés dans le `Terminal`.

4.2 Redirections de flux

Le `Shell` vous permet de rediriger à votre convenance les flux d'informations :

— d'un programme vers un fichier afin de récupérer les flux de sortie du programme dans le fichier en question plutôt que dans le terminal,

— depuis un fichier vers un programme, le fichier contenant le flux d'entrée, plutôt que de le lui donner par le terminal,

— entre deux programmes, pour envoyer comme flux d'entrée au second le flux de sortie du premier.

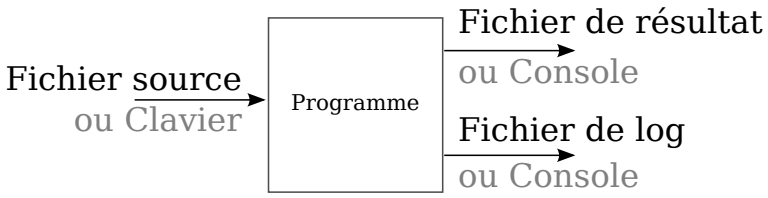

Figure 7 – ***Nous pouvons rediriger les flux d'un programme vers ou depuis des fichiers.***

Voici les opérateurs pour rediriger ces flux. Ceux marqués d'une étoile * sont **à retenir absolument**.

TABLE 8: Opérateurs `Shell` liés à la redirection.

Opérateur	Commande	Effet
`>`*	`CMD > FILE`	Redirige la sortie de CMD vers le fichier FILE (Écrase le fichier préexistant).
`>>`*	`CMD >> FILE`	Redirige la sortie de CMD vers le fichier FILE (Ajoute à la fin du fichier).
`2>`	`CMD 2> FILE`	Redirige le flux d'erreur de CMD vers le fichier FILE (Écrase le fichier préexistant).
`2>>`	`CMD 2>> FILE`	Redirige le flux d'erreur de CMD vers le fichier FILE (Ajoute à la fin du fichier).
`<`*	`CMD < FILE`	Redirige le fichier FILE sur le flux d'entrée de CMD (À la place du clavier).
`|`*	`CMD1 | CMD2`	Redirige la sortie de CMD1 vers le flux d'entrée de CMD2.

Comme nous pouvons le voir sur la figure 7, les opérateurs de redirection vers des fichiers (chevron(s) vers la droite) permettent de sauvegarder les résultats de l'opération dans un fichier pour une étude ultérieure par exemple. Ils sont beaucoup utilisés pour le suivi d'erreurs (fichier de log) ou la sauvegarde de résultats lors de longs calculs.

EXERCICE 19

Lancez la commande `ls`. Puis lancez la commande `ls > result.txt`. Affichez le contenu du fichier `result.txt` (cat). Que s'est-il passé ?

EXERCICE 20

Relancez la commande `ls > result.txt`. Affichez à nouveau le contenu de ce fichier. Voyez-vous un changement ? Testez `ls » result.txt` maintenant. Affichez à nouveau le fichier. Que s'est-il passé ?

EXERCICE 21

Avec la commande `mkdir`, créez dans le dossier « Bureau » le dossier « Test ». Relancez la même commande. Normalement vous obtenez un message d'erreur. Lancez `mkdir Test > log.txt`. Avez-vous toujours le message qui s'affiche dans le terminal ? Affichez le contenu du fichier `log.txt`, qu'y a-t-il dedans ? Lancez maintenant la commande `mkdir Test 2> log.txt`. Que se passe-t-il ? Affichez le contenu du fichier `log.txt`.

Remarque

Sous `Unix`, il existe un fichier spécial `/dev/null` qui correspond... au néant ! Il permet de ne pas afficher les messages qui ne vous intéressent pas.

```
ls toto.txt 2> /dev/null
```

Quand nous utilisons un fichier pour donner le flux d'entrée à un programme, cela nous permet de mettre facilement une grande quantité d'informations dedans : c'est moins fastidieux que de les taper dans la console. De plus, nous pouvons renvoyer ce même flux autant de fois que nous le souhaitons et de manière identique.

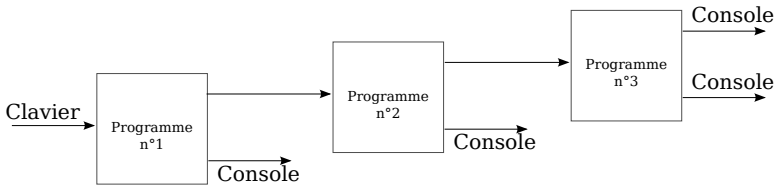

Figure 8 – ***Nous pouvons chaîner plusieurs programmes.***

Grâce à l'opérateur de chaînage `|` (tube ou pipe en anglais, obtenu par la combinaison `ALTGR+6`), nous pouvons relier la sortie d'un programme avec l'entrée du suivant. Nous pouvons alors construire une chaîne de programmes qui font chacun une partie du travail (cf. figure 8) : mise en forme, filtrage, tri, etc.

C'est pourquoi les systèmes UNIX sont munis de tout un arsenal de petits utilitaires de manipulation des flux d'informations. Les commandes marquées d'une étoile * sont **à retenir absolument**.

TABLE 9: Commandes `Shell` liées à la manipulation de flux.

Nom	Commande	Effet
MORE	`CMD | more`	L'affichage se restreint à la taille de votre terminal, et une action de votre part est attendue pour afficher la suite (tampon). Nous ne pouvons pas remonter.
LESS*	`CMD | less`	Comme more, mais nous pouvons remonter dans l'affichage (pagination).
HEAD	`CMD | head -n N`	N'affiche que les N premières lignes du flux d'informations.
TAIL	`CMD | tail -n N`	N'affiche que les N dernières lignes du flux d'informations.
GREP	`CMD | grep PATTERN*`	Filtre le flux d'information et ne garde que les lignes qui contiennent PATTERN.
	`grep PATTERN FILE`	Affiche les lignes du fichier qui contiennent PATTERN.
SORT	`CMD | sort`	Trie les lignes du flux d'informations. (man sort pour plus d'info.)
	`sort FILE1 FILE2`	Affiche les lignes triées des fichiers FILE1 FILE2. (man sort pour plus d'info.)
Word Count	`CMD | wc -l`	Renvoie le nombre de lignes du flux d'informations.
	`CMD | wc -w`	Renvoie le nombre de mots du flux d'informations.

(Suite page suivante)

Nom	Commande	Effet
	`CMD | wc -c`	Renvoie le nombre de caractères du flux d'informations.
CUT	`CMD | cut -d DEL -f N`	Pour chaque ligne du flux d'informations, renvoie le Nième fragment délimité par DEL.

QUESTION 15


Comment récupéreriez vous **le nombre** d'utilitaires contenus dans le dossier `/bin/` qui commence par la lettre `g` ?

QUESTION 16 (*Défis*[1])


Sachant que les informations concernant le(s) processeur(s) sont stockées dans le fichier `/proc/cpuinfo`, quelle chaîne de commande faut-il taper pour afficher uniquement le nombre de processeurs de la machine (plusieurs réponses possibles) ?

Mémo

« Linux est un noyau, GNU est un ensemble de programmes utilitaires et les deux ensembles forment un système d'exploitation. »
« Tout ce que vous faites sur l'interface graphique peut se faire en commandes Shell. »
« Le terminal, c'est très rapide et très efficace. »
« Les commandes ne s'apprennent pas par coeur mais par la pratique ! »
« En chainant les commandes, nous pouvons effectuer des tâches plus complexes. »

1. Les questions Défis sont des questions optionnelles que vous pouvez faire à la maison pour approfondir le TD.