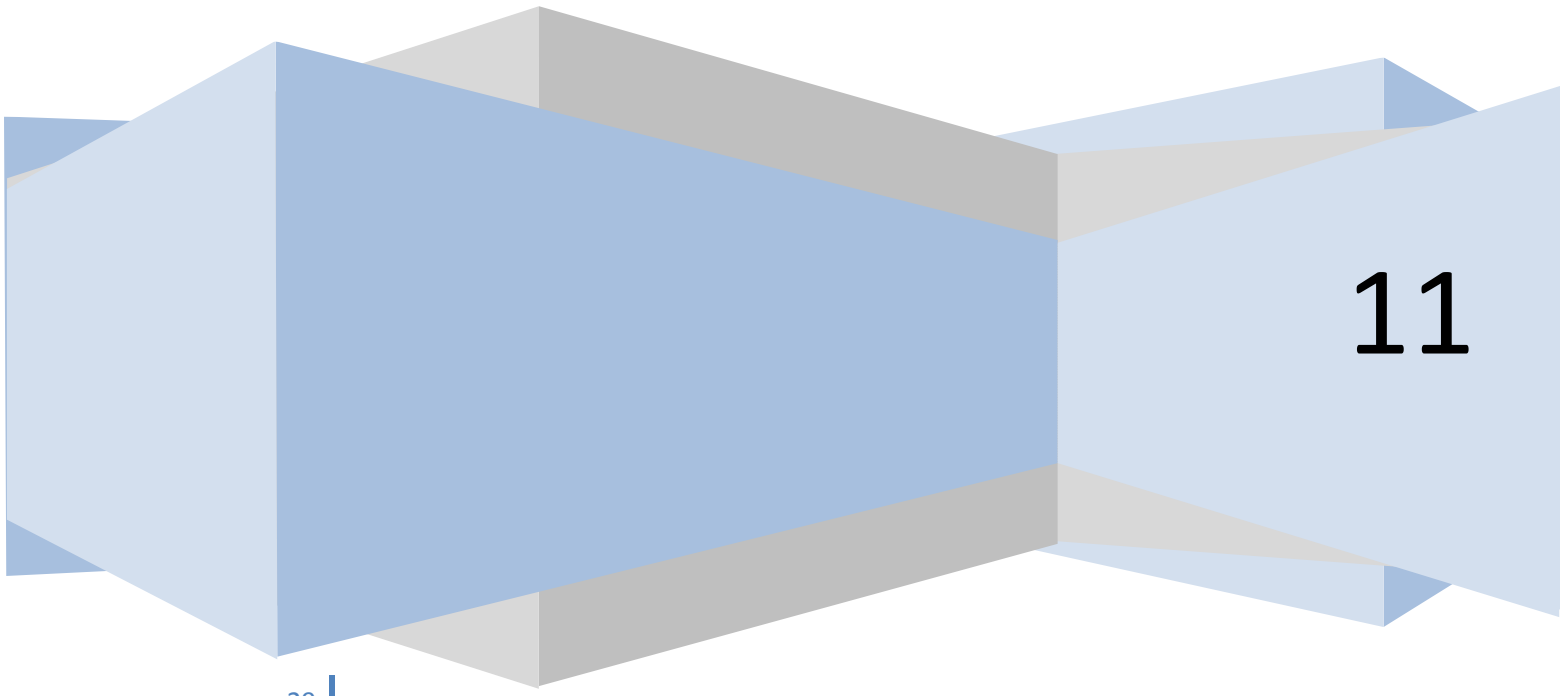
EFREI

TD Unix pour l'utilisateur

Partie 2

tpunix@efrei.fr



11

Les réponses aux exercices devront être écrites dans un fichier texte - pas de document Open Office ou de document Word, un simple fichier texte - dont le nom sera composé de :

- La lettre désignant votre groupe
- Les noms des membres de votre binôme
- La date

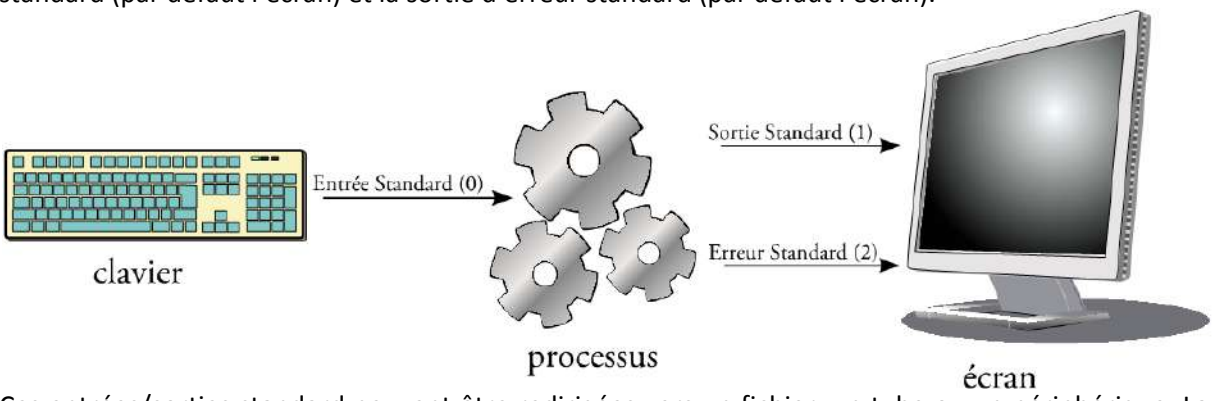
Voici un exemple de nom de fichier :
`B_edward_alphonse_19052008.txt`
N'hésitez pas à illustrer votre compte rendu avec des copiés/collés venant directement de votre terminal (pas des images, seulement le texte).
N'oubliez pas, pour chaque question, de préciser son numéro.

A la fin du TP, il conviendra de m'envoyer ce fichier par mail à l'adresse *vinzelle@efrei.fr*, en précisant le nom du fichier dans le champ 'Sujet' du mail.

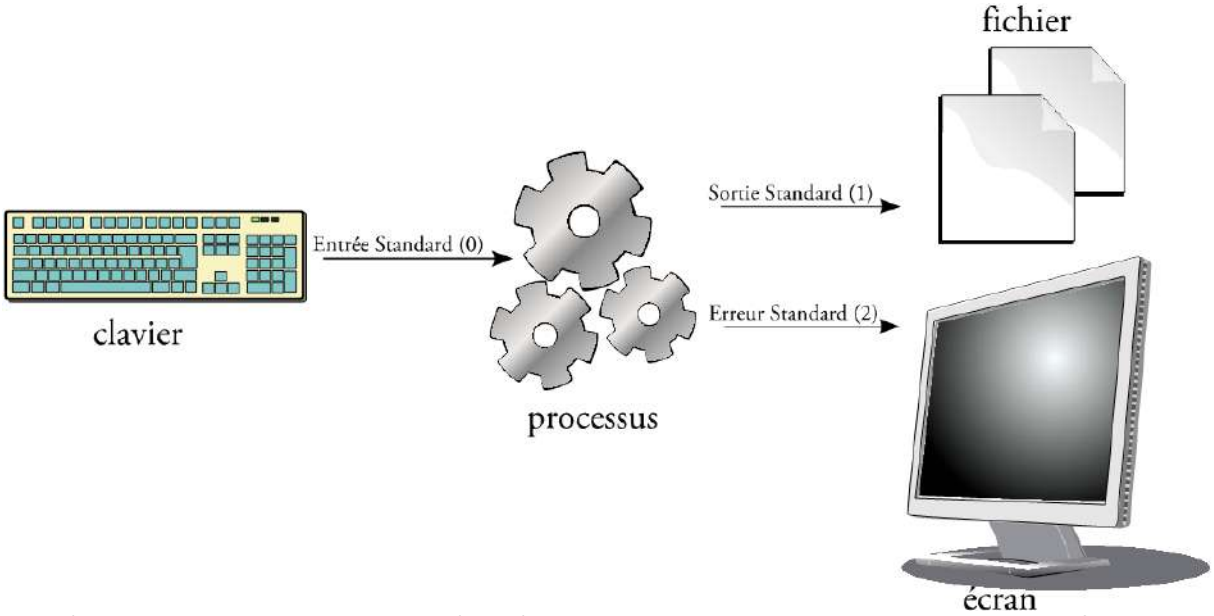
6 Les redirections, pipes et filtres

6.1 Les redirections

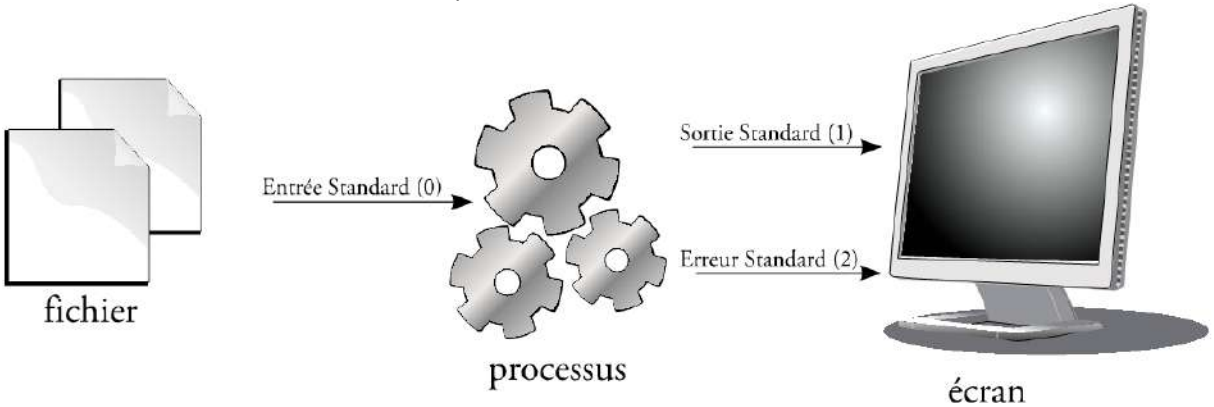
On appelle processus, ou tâche, l'exécution d'un programme. Au lancement de chaque processus, l'interpréteur de commandes ouvre d'office une entrée standard (par défaut le clavier), une sortie standard (par défaut l'écran) et la sortie d'erreur standard (par défaut l'écran).



Ces entrées/sorties standard peuvent être redirigées vers un fichier, un tube ou un périphérique. La redirection de la sortie standard vers un fichier consiste à renvoyer le texte qui apparaît à l'écran vers un fichier.



Il est également possible de rediriger l'entrée standard, le processus cherchera alors les informations dont il a besoin dans un fichier et non plus sur le clavier.



Le caractère < suivi d'un nom de fichier indique la redirection de l'entrée standard à partir de ce fichier : `<fe`, avec `fe` le fichier d'entrée.

Le caractère > suivi du nom d'un fichier indique la redirection de la sortie standard vers ce fichier : `> fs`, avec `fs` le fichier de sortie. Le fichier `fs` est créé ou écrasé s'il existe déjà.
Si on double le caractère >, comme dans `>> fs`, l'information ou la redirection sera ajoutée au fichier `fs`.
La sortie standard est désignée par le descripteur de fichier 1, et l'erreur standard est désignée par le descripteur de fichier 2.
Si l'on souhaite rediriger la sortie standard vers un fichier, et ne pas modifier l'erreur standard, on utilise : `1> fs`. De même si l'on ne veut rediriger que l'erreur standard vers un fichier, on utilise : `2> fs`.
Dans le cas où l'on souhaite rediriger les deux sorties standards vers un fichier, on écrit : `&> fs`.
Enfin, pour rediriger le flux d'erreur standard vers le flux de sortie standard, on utilise la syntaxe `2>&1`.
Voici quelques exemples :

- Redirection de la sortie standard vers un fichier.

```
vinzelle@pommard:~$ ls
Mail  News  rep1  testdir  Unix
vinzelle@pommard:~$ ls > temp
vinzelle@pommard:~$ cat temp
Mail
News
rep1
temp
testdir
Unix
```

- Redirection de la sortie standard et de l'erreur standard dans un fichier.

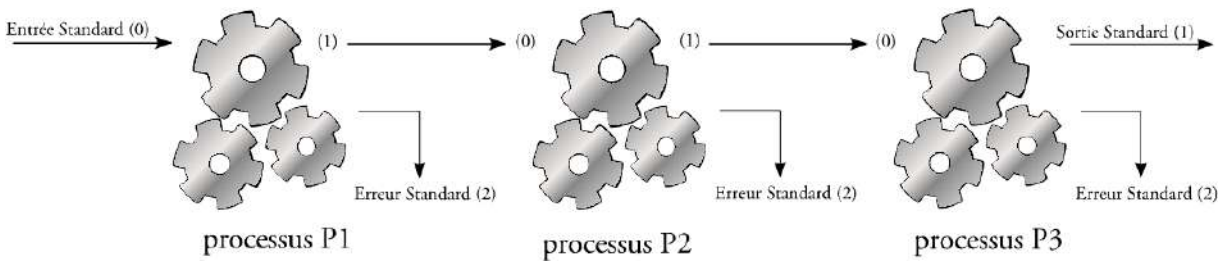
```
vinzelle@pommard:~$ ls ./rep1 ./rep2
ls: ./rep2: Aucun fichier ou répertoire de ce type
./rep1:
fich11  fich12  rep2  rep3
vinzelle@pommard:~$ ls ./rep1 ./rep2 &> temp
vinzelle@pommard:~$ cat temp
ls: ./rep2: Aucun fichier ou répertoire de ce type
./rep1:
fich11
fich12
rep2
rep3
```

- Redirection de la sortie standard et de l'erreur standard dans deux fichiers différents.

```
vinzelle@pommard:~$ ls ./rep1 ./rep2 2> temp.error 1> temp.list
vinzelle@pommard:~$ cat temp.list
./rep1:
fich11
fich12
rep2
rep3
vinzelle@pommard:~$ cat temp.error
ls: ./rep2: Aucun fichier ou répertoire de ce type
```

6.2 Les tubes

Un tube, ou plus communément « pipe » en anglais, est un flot de données qui permet de relier la sortie standard d'une commande à l'entrée standard d'une autre commande sans passer par fichier temporaire.



Dans une ligne de commande, le tube est formalisé par la barre vertical, nommé pipe et noté |. Le schéma précédent peut se décrire ainsi sur une ligne de commande :

```
P1 | P2 | P3
```

Voici quelques exemples de tubes couramment utilisés :

```
vinzelle@pommard:/$ ls -l | more
```

Dans cet exemple, le résultat de la commande ls -l n'apparait pas à l'écran : la sortie standard est redirigée vers l'entrée standard de la commande more qui, quant à elle, affichera son entrée standard page par page.

```
vinzelle@choam:~$ who | wc -l
14
```

Cette commande permet de compter le nombre d'utilisateurs connectés actuellement.

Attention : beaucoup de commandes ne lisent malheureusement pas dans leur entrée standard, ce qui rend impossible leur assemblage avec des tubes. Par exemple les commandes ls, rm, cp, ln, mv et bien d'autres ne sont pas des filtres, mais traitent leurs arguments :

```
rm fich1 fich2 fich3
```

est syntaxiquement correct, alors que :

```
ls | rm
```

n'a aucun sens.

Pour contourner ce problème nous avons à notre disposition la commande `xargs` :

```
cmd1 | xargs cmd2
```

signifie « lancer cmd2 en lui passant en paramètres ce qui arrive dans l'entrée standard de `xargs` », donc la sortie standard de cmd1.

6.3 Les filtres

Les filtres sont une utilisation particulièrement astucieuse des tubes.

En effet, un filtre est une commande qui lit les données sur l'entrée standard, les traite et les écrit sur la sortie standard.

Le concept de tube devient un outil très puissant dans Unix qui propose un choix très vaste de filtres.

Voici une liste non exhaustive de filtres couramment utilisés :

- `grep` & `egrep` : recherche les occurrences d'une chaîne de caractère (cf 5.6.6)
- `wc` : compte le nombre de caractères, mots et lignes (cf 5.6.7)
- `more` & `less` : affiche son entrée standard par page écran (cf 5.6.1 & 5.6.2)
- `sed` : éditeur de flot, applique les commandes de l'éditeur ed sur l'entrée standard et envoie le résultat sur la sortie standard
- `awk` : petit langage de manipulation de texte
- `sort` : filtre de tri

Voici un exemple illustrant bien la puissance de tubes et des filtres. Il s'agit de compter le nombre d'utilisateur de la machine dont le shell est le *bash*.

```
skyce@samvimes ~ $ cat /etc/passwd | grep /bin/bash | wc -l
8
```

6.4 Exercices

6.4.1 Exercice

Le fichier `/etc/services` contient la liste des services réseaux Unix. Quelle commande vous permet d'afficher page par page toutes les lignes du fichier `/etc/services` qui contiennent un ou plusieurs dièses ?

6.4.2 Exercice

Affichez page par page les lignes de `/etc/services` qui ne contiennent que du commentaire (lignes commençant par un dièse). Donnez la commande que vous avez utilisée.

6.4.3 Exercice

Affichez le fichier `/etc/services` trié dans l'ordre numérique décroissant et selon le deuxième champ. Donnez la commande utilisée.

6.4.4 Exercice

La commande `find ~ | xargs grep "#"` permet de lister les fichiers de votre dossier d'utilisateur qui contiennent un dièse mais elle génère des messages d'erreurs pour les répertoires et pour certains fichiers comportant des espaces dans leurs noms. Proposez une commande qui permet de compter les lignes du flux d'erreur.

6.4.5 Exercice

Comment pouvez-vous avec la commande `tee` rediriger la sortie de la commande `ping localhost` dans le fichier ping.out tout en visualisant dans la console la sortie en temps réel ?

Note : pensez à consulter la commande `man tee`

6.4.6 Exercice

Proposez un filtre `grep` qui permette d'afficher 5 lignes avant et 5 lignes après la ligne contenant le mot ftp du fichier /etc/services, tout en affichant le numéro de chaque ligne.

6.4.7 Exercice

Proposez un filtre `grep` qui permette d'afficher le nombre d'occurrences par fichier du mot user dans tout le répertoire /etc et ses sous répertoires. Prenez soin de ne pas afficher les erreurs.

6.4.8 Exercice

Testez la ligne de commande suivante : `wc << EOF`

Comment fonctionne-t-elle ?

7 Les variables d'environnements, les alias et le prompt

7.1 Les principales variables d'environnement

Toutes les variables d'environnements de l'environnement Unix sont précédées du caractère « $ » et encadrées par des accolades : ${VARIABLE}. Pour avoir la valeur d'une variable il suffit de taper la commande `echo $VARIABLE`.

Voici une liste non exhaustive des principales variables d'environnement sur les systèmes Unix :

Variable d'environnement	Description
$ARCH	Contient la description de l'architecture de la machine.
$DISPLAY	Contient l'identifiant du terminal d'affichage à utiliser dans le gestionnaire de fenêtres (X11).
$HOME	Retourne le chemin d'accès vers le répertoire de l'utilisateur courant.
$HOST	Retourne le nom de l'ordinateur.
$LANG	Retourne le code de langue par défaut.
$PATH	Retourne une liste de chemins d'accès vers des répertoires contenant les

	exécutables, séparés par des points-virgules.
$PRINTER	Contient le nom de l'imprimante par défaut.
$SHELL	Indique le chemin de l'interpréteur de commande utilisé.
$USER	Retourne l'identifiant de l'utilisateur courant.

7.2 La variable $PATH

Le premier champ d'une commande désigne le nom de la commande elle-même.
Si ce n'est pas une commande interne, les shells essaient de trouver un fichier exécutable qui porte ce nom. Toutefois ce fichier n'est cherché que dans certains répertoires: ceux dont les noms sont listés dans la variable d'environnement $PATH et dans l'ordre où ils s'y trouvent.
Vous pouvez voir le contenu de cette variable avec la commande suivante:

```
vinzelle@pommard:~$ echo $PATH
/users/guest/vinzelle/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
```

Ainsi lorsque vous tapez `ls`, c'est usuellement le fichier `/bin/ls` qui est exécuté, mais tout dépend du contenu de la variable $PATH et des binaires disponibles sur le système.
Il est possible de forcer l'exécution d'un binaire particulier en indiquant le chemin absolu vers ce binaire.
La commande `which` permet de savoir quel est le fichier exécuté pour une commande donnée.

```
vinzelle@pommard:~$ /usr/bin/who
vinzelle pts/0        2008-05-17 13:31 (choam.etudiants.efrei.fr)
vinzelle@pommard:~$ which who
/usr/bin/who
vinzelle@pommard:~$ who
vinzelle pts/0        2008-05-17 13:31 (choam.etudiants.efrei.fr)
```

7.3 Alias

Les alias permettent de définir de nouvelles commandes ou de modifier le comportement de commandes existantes. C'est un mécanisme plus léger que l'utilisation de scripts en combinaison avec le positionnement de la variable $PATH. De plus, ce n'est qu'avec ce mécanisme d'alias que l'on peut modifier le comportement des commandes internes aux shells.
Le principe de fonctionnement est celui d'une macro qui ne travaille que sur le premier champ des commandes.

```
vinzelle@pommard:~$ ll
-bash: ll: command not found
vinzelle@pommard:~$ alias ll='ls -la'
vinzelle@pommard:~$ ll
total 41
drwx--x--x  10 vinzelle vacat  512 2008-05-12 14:25 .
drwxr-xr-x 163 root     root  3072 2008-04-03 15:47 ..
-rw-------   1 vinzelle vacat 6399 2008-05-16 21:25 .bash_history
-rw-r--r--   1 vinzelle vacat    6 2008-04-03 15:47 .bash_logout
-rw-r--r--   1 vinzelle vacat  177 2008-04-03 15:47 .bash_profile
-rw-r--r--   1 vinzelle vacat  460 2008-04-03 15:47 .bashrc
drwx------   2 vinzelle vacat  512 2008-04-08 14:38 .gnupg
…
```

De plus la commande `alias` permet de lister les alias.

```
vinzelle@pommard:~$ alias
alias ll='ls -la'
```

7.4 Le prompt ou Invite de Commande

Le shell s'initialise en lisant sa configuration globale (dans un fichier du répertoire `/etc/`), puis en lisant la configuration propre à l'utilisateur (dans un fichier caché, dont le nom commence par un point, situé dans le répertoire de base de l'utilisateur, c'est-à-dire `/home/nom_de_l_utilisateur/.fichier_de_configuration`), puis il affiche une invite de commande (en anglais *prompt*) comme suit :

```
machine:/repertoire/courant$
```

Par défaut, dans la plupart des shells, le prompt est composé du nom de la machine, suivi de deux points (:), du répertoire courant, puis d'un caractère indiquant le type d'utilisateur connecté:

- «$» indique qu'il s'agit d'un utilisateur normal
- «#» indique qu'il s'agit de l'administrateur, appelé «root»

La variable d'environnement `$PS1` est affichée systématiquement comme prompt et `$PWD` contient "en temps réel " la valeur du répertoire courant, en procédant ainsi :

```
vinzelle@pommard:~$ export PS1="$(echo 'machine:$PWD>')"
```

le prompt Unix aura cette apparence :

```
machine:/users/guest/vinzelle>
```

7.5 La substitution

Il est possible de faire prendre à une variable la valeur de la sortie standard d'une commande avec cette syntaxe toute simple :

```
vinzelle@pommard:~$ export TEST=`who am i`
vinzelle@pommard:~$ echo $TEST
vinzelle pts/0 2008-05-18 16:39 (choam.etudiants.efrei.fr)
```

7.6 Exercices

7.6.1 Exercice

Que fait la commande `printenv` ?

7.6.2 Exercice

Attribuez la valeur de votre choix à la variable du shell `$VARIABLE`. Affichez ensuite le contenu de cette variable. Supprimez ensuite cette variable.

7.6.3 Exercice

Stockez la date du jour dans une variable. Affichez ensuite cette variable.

7.6.4 Exercice

Affichez puis sauvegardez le contenu de la variable `$PATH` dans un fichier nommé *variable.path*.

7.6.5 Exercice

Que faut-il faire pour que vous puissiez exécuter le fichier `bienvenue.sh` de la question 5.7.6 depuis n'importe quel répertoire ?

7.6.6 Exercice

Créez un tableau de variables contenant « pomme », « poire », « abricot ».

Donnez la commande permettant d'afficher « abricot ».

Substituez « abricot » par « saucisson », puis affichez le tableau au complet.

7.6.7 Exercice

Créez dans votre session un alias nommé `la` qui vous permet de lister de façon récursive les fichiers du répertoire courant et de ses sous-répertoire au format long en incluant les fichiers cachés. Quelle commande avez-vous utilisée ? Quelle commande vous permet de lister tous les alias de votre environnement ?

7.6.8 Exercice

Donnez la commande qui permet de positionner le prompt à : "[login@machine heure]$ ". Utilisez le même formatage de date que celui proposé dans cet exemple :

```
[vinzelle@pommard 22:33:55]$
```

7.6.9 Exercice

En modifiant votre fichier *.bashrc,* activez l'alias `ll`, puis activez le sans ouvrir un nouveau shell (indice : la commande source).

8 L'éditeur de texte vi

8.1 Présentation de l'éditeur

L'éditeur vi [à prononcer « *vi-aïe* »] est l'éditeur standard du système d'exploitation UNIX. C'est un éditeur modal, c'est-à-dire qu'il est toujours dans l'un ou l'autre des trois modes disponibles. Selon le mode dans lequel il se trouve, l'éditeur interprète différemment les caractères qu'il reçoit du clavier :

- Le mode **normal**: celui dans lequel vous êtes à l'ouverture du fichier. Il permet de taper des commandes
- Le mode **insertion**: Ce mode permet d'insérer les caractères que vous saisissez à l'intérieur du document. Pour passer en mode insertion, il suffit d'appuyer sur la touche Insert de votre clavier, ou à défaut de la touche i
- Le mode de **remplacement**: Ce mode permet de remplacer le texte existant par le texte que vous saisissez. Il vous suffit de ré-appuyer sur r pour passer en mode remplacement, et d'appuyer sur la touche Echap pour revenir en mode normal

Il existe un très grand nombre de commandes pour les différents modes. Voici les plus courantes (les autres seront facilement retrouvées sur internet) :

Les commandes de base du mode normal :

:q! Entrée	Pour quitter sans sauver
:w Entrée	Pour enregistrer
:wq Entrée	Pour enregistrer et quitter
:u Entrée	Permet d'annuler (ou :undo)
x	Efface le caractère qui se trouve sous le curseur
4x	Efface les 4 caractères à partir de la position du curseur
dd	Efface la ligne sur laquelle se trouve le curseur
6dd	Efface les 6 lignes à partir de la position du curseur
rx	Remplace le caractère courant par le caractère x
s	Remplace le caractère courant (*substitution*)
cw	Remplace le mot courant ou suivant (*change word*)
cb	Remplace le mot courant ou précédent
S ou cc	Remplace tous les caractères de la ligne courante
C	Remplace la fin de la ligne courante (depuis le curseur)

Les commandes de base du mode insertion :

a	Ajouter du texte après le curseur
A	Ajouter du texte en fin de ligne courante
i	Insérer du texte avant le curseur
I	Insérer du texte en début de ligne
o ou O	Créer une ligne vierge sous ou au dessus du curseur
R	Remplacer le texte à partir du curseur
n[s]	Supprime n caractères et passe en mode insertion
Ctrl+h ou Backspace	Efface la lettre précédente
Ctrl+w	Efface le mot précédent
CTRL+v Ctrl+M	Pour écrire ^M

Pour clore ce chapitre, une petite astuce bien pratique si vous utilisez `vim`, essayez la commande `:syntax on`. Cela active la coloration syntaxique qui peut s'avérer bien pratique lorsque l'on édite des scripts.

8.1.1 Exercice

Saisir dans un fichier boetie.txt les quatre lignes suivantes d'Étienne de La Boétie à l'aide de l'éditeur vi. Créez ensuite une copie de ce fichier que vous nommerez `boetie2.txt`. Vous

effectuerez les traitements demandés sur `boetie2.txt`. Le fichier `boetie.txt` vous servira de sauvegarde de secours en cas de commande destructrice.

Note : Vous êtes dispensés de saisir les accents si vous travaillez sur un clavier qwerty.

J'en sçay le vray, et si cest esprit là
Se laissoit voir avecques sa grandeur,
Alors vrayment verroit l'on par grand heur
Les traicts, les arcs, les amours qui sont là.

Indiquez toutes les commandes que vous avez utilisées pour saisir le texte et pour le sauvegarder.

Testez sur `boetie2.txt` la commande vi `:%s/[,.']//g`

Que fait cette commande ?

Proposez une commande qui permet de substituer toutes les occurrences de ay en ai.

Une fois toutes ces modifications effectuées, sauvegarder le fichier `boetie2.txt`.

8.1.2 Exercice

Pour les exercices suivants, vous pouvez utiliser le fichier texte se trouvant sur http://perso.efrei.fr/~vinzelle/disclaimer.txt.

Comment se rendre à la dernière ligne du buffer ? Comment revenir ensuite à la première ?

8.1.3 Exercice

Comment supprimer les 5 premières lignes du document ?

8.1.4 Exercice

Comment copier/coller une ligne ?

8.1.5 Exercice

Comment copier les 5 dernières lignes du document ?

8.1.6 Exercice

Comment fonctionnent les commandes *cw* et *y$* ?

9 Commandes relatives à la gestion de processus

9.1 Tâches en avant plan et arrière plan

Unix attend la fin de l'exécution de la commande en cours avant de permettre à l'utilisateur de relancer une nouvelle commande. Lorsqu'une commande ne nécessite pas de dialogue avec l'utilisateur et que sa durée d'exécution est importante, il est possible de l'exécuter en arrière plan en ajoutant le caractère spécial `&` à la fin de la commande. Le système Unix lance alors la commande et redonne immédiatement la main à l'utilisateur pour d'autres travaux.
Le processus n'étant plus connecté à un terminal, il devient alors impossible de l'interrompre avec ctrl+d, pour terminer ce processus il faut alors :

- sortir de sa session, ce qui a pour effet de tuer tous les processus qui lui sont attachés (un `ps f` pour s'en convaincre !)
- rechercher le numéro du processus et lui envoyer un signal à l'aide de la commande `kill`

- ramener le processus en avant plan avec la commande `fg <numéro du processus>`, le numéro du processus pouvant s'obtenir avec la commande `jobs`, on peut alors le stopper momentanément avec la combinaison de touches ctrl+z ou définitivement avec ctrl+c. Si le processus est momentanément stoppé, il peut être relancé avec la commande `bg <numéro du processus>`.

Voici un exemple d'utilisation de ces commandes :

```
skyce@samvimes ~ $ jobs
skyce@samvimes ~ $ ./scripts/loop.sh &
[1] 32083
skyce@samvimes ~ $ jobs
[1]+  Running                 ./scripts/loop.sh &
skyce@samvimes ~ $ fg 1
./scripts/loop.sh
ctrl+z
[1]+  Stopped                 ./scripts/loop.sh
skyce@samvimes ~ $ bg 1
[1]+ ./scripts/loop.sh &
skyce@samvimes ~ $ jobs
[1]+  Running                 ./scripts/loop.sh &
skyce@samvimes ~ $ fg 1
./scripts/loop.sh
ctrl+c
skyce@samvimes ~ $ jobs
```

9.2 Lister les processus, évaluer la charge de la machine

9.2.1 La commande ps

La commande `ps` permet d'obtenir des renseignements sur l'état des processus en cours. Par défaut, seuls les processus de la connexion en cours pour l'utilisateur ayant lancé la commande `ps` sont affichés. L'option `-u <username>` permet de sélectionner les processus de cet utilisateur pour l'ensemble de ses connexions.

Voici un exemple :

```
vinzelle@pommard:~$ ps -u vinzelle
  PID TTY          TIME CMD
14193 ?        00:00:00 sshd
14196 pts/0    00:00:00 bash
14508 pts/0    00:00:00 ps
```

Pour chaque processus, il est ainsi possible de connaître, avec les options adéquates :

- le nom de l'utilisateur (UID)
- le numéro du processus (PID)
- le numéro du processus père (PPID)
- l'heure de lancement du processus (STIME)
- le nom du terminal (TTY)
- le temps d'exécution du processus (TIME)

9.2.2 La commande top

La commande `top` est l'équivalent interactif de `ps`.

Cette commande affiche à l'écran l'ensemble des processus en cours d'exécution, tout en donnant leur état, la taille de leur occupation mémoire, la charge CPU, des statistiques systèmes sur la mémoire, la charge… A l'aide de `top` il est également possible d'envoyer les signaux de son choix aux processus. C'est une commande très utile et complète, dont voici un exemple :

```
top - 23:16:02 up 27 days,  8:43,  1 user,  load average: 0.01, 0.02, 0.00
Tasks:  80 total,   1 running,  79 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.1%us,  0.1%sy,  0.0%ni, 99.5%id,  0.0%wa,  0.0%hi,  0.2%si
Mem:    513788k total,   502392k used,    11396k free,        0k buffers
Swap:   995988k total,      624k used,   995364k free,   169992k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      20   0  1944  656  564 S  0.0  0.1   0:13.68 init
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kthreadd
```

```
    3 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      15  -5     0    0    0 S  0.0  0.0   0:01.90 ksoftirqd/0
    5 root      15  -5     0    0    0 S  0.0  0.0   0:45.27 events/0
    6 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
   60 root      15  -5     0    0    0 S  0.0  0.0   0:00.90 kblockd/0
   63 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   64 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kacpi_notify
  135 root      15  -5     0    0    0 S  0.0  0.0   4:58.55 ata/0
  136 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 ata_aux
  138 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
  168 root      20   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  169 root      20   0     0    0    0 S  0.0  0.0   0:12.22 pdflush
  170 root      15  -5     0    0    0 S  0.0  0.0   0:24.00 kswapd0
  171 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 aio/0
  172 root      15  -5     0    0    0 S  0.0  0.0   0:00.10 cifsoplockd
```

Une indication de la charge d'une machine est le "load average" qui se compose de 3 chiffres.

```
load average: 0.01, 0.02, 0.00
```

Le premier chiffre correspond au nombre de processus traités par le processeur en moyenne par minute, le deuxième chiffre correspond au nombre de processus traité par le processeur en moyenne pour cinq minutes, et le dernier chiffre correspond enfin à cette valeur calculée sur quinze minutes.

- On suppose qu'une machine mono processeur n'est pas chargée si sa charge est < 1. Cela signifie qu'il n'y a pas de compétition entre les processus, et que le processeur n'est pas sollicité en permanence. Pour améliorer les performances, il est peut être nécessaire de paralléliser le traitement, et/ou d'améliorer la vitesse des entrées/sorties (disque, réseau, mémoire).
- On suppose qu'une machine mono processeur est pleinement utilisée quand sa charge est en permanence à 1. Pour améliorer ses performances, on peut ajouter un processeur plus puissant, et s'assurer qu'il n'y a pas de goulet d'étranglement au niveau entrées/sorties.
- Si une machine mono processeur a une charge > 1, on suppose qu'elle est très chargée. Une fois encore, il faut bien s'assurer que le problème est un problème de puissance de calcul et pas seulement un problème de vitesse d'entrée/sortie.

Pour pouvoir faire une étude plus fine de la charge d'une machine, il faut alors s'intéresser aux autres chiffres statistiques proposées par `top`, et notamment la ligne `Cpu (s)`.

On peut remarquer que l'on bénéficie de plusieurs indicateurs pour juger de la charge d'une machine. Ces différents indicateurs sont :

```
Cpu(s):  0.1%us,  0.1%sy,  0.0%ni, 99.5%id,  0.0%wa,  0.0%hi,  0.2%si
```

Ces indicateurs correspondent à:

- User - le temps consommé par les processus utilisateurs
- System - le temps consommé par les processus dépendant du kernel
- Nice - le temps consommé par les processus utilisateurs rendus prioritaires
- Idle - le temps passé à ne rien faire
- Wait - le temps consommé à attendre des entrées/sorties
- Hardware IRQ - le temps consommé à gérer des interruptions matérielles
- Software Interrupts - le temps consommé à gérer des interruptions logicielles

Ces différents indicateurs permettent de proposer un diagnostique complet sur l'état de charge de la machine, et les différentes améliorations que l'on peut apporter au système pour qu'il traite correctement ses tâches.

9.3 Arrêter un processus

La commande `kill` sert à arrêter des processus, ou quand ils sont plantés, à tuer des processus.
Un certain nombre d'évènements peuvent être signalés à un processus à l'aide de signaux. Ce mécanisme permet la communication interprocessus, notamment entre le système d'exploitation et le processus.
Les principaux signaux sont :

SIGHUP	SIGnal Hang UP : fin du shell
SIGINT	SIGnal INTerrupt : interruption du programme
SIGQUIT	SIGnal QUIT : terminaison brutale
SIGKILL	SIGnal KILL : tuer le processus
SIGTERM	SIGnal TERMinate : terminaison douce

Par défaut, la commande `kill` envoie un signal SIGTERM.

9.4 Lister les fichiers ouverts par un processus

Une commande très pratique quand on travaille sur des problèmes de processus bloqués, en attente de l'accès à un fichier par exemple, est la commande `lsof`. Elle permet en effet de lister l'ensemble des ressources systèmes utilisées par des processus (mémoire, socket, fichier, périphérique, terminal…).

9.5 Exercices

9.5.1 Exercice

Affichez l'ensemble des signaux que sait envoyer kill. Quel est la syntaxe la plus courte pour envoyer un signal SIGKILL à un processus ?

Quelle est la commande qui permet de tuer un processus par son nom, et non son PID ?

9.5.2 Exercice

Quelle est la différence entre SIGTERM et SIGKILL ?

9.5.3 Exercice

Citez cinq différents états de processus identifiables avec la commande `ps axw`.

Note : pensez à consulter la commande `man ps`

9.5.4 Exercice

Comment obtenir la liste des processus en cours d'exécution en temps réel ?

9.5.5 Exercice

Dans un shell, lancez la commande `xterm`

1. Dans un autre shell, utilisez la commande `ps` pour connaître son état. Quel est son état ?
2. Revenez dans le premier shell et tapez la séquence *Ctrl-Z*. Que s'est il passé ? Quel état vous indique la commande `ps` ?
3. Comment faire pour réactiver le processus `xterm`, tout en le laissant en arrière plan ?
4. Lancez maintenant `xterm` en tâche de fond. Donnez la commande utilisée.
5. Quelles commandes vous permettent maintenant de repasser `xterm` en avant plan ?

9.5.6 Exercice

A l'aide de la commande `lsof`, afficher la liste des fichiers manipulés, des terminaux ouverts, et des librairies chargées par le processus `bash`.

9.5.7 Exercice

Lancer `xterm` en tâche de fond de manière à ce qu'elle soit détachée de son terminal de contrôle (utilisez la commande `nohup`). Quitter votre shell pour vérifier que `xterm` ne disparaît pas. Ouvrir un nouveau shell puis récupérer l'identifiant du processus et le tuer avec la commande `kill`. Indiquer toutes les commandes utilisées ainsi que le nom du fichier à effacer après la manipulation.