Petit Guide de Survie sous UNIX

Contents

1. Introduction

Ce petit livret a été écrit pour permettre aux novices d'arriver à passer le cap de la connexion au réseau et l'exécution de quelques commandes simples. C'est à leur intention que l'essentiel de ce manuel est consacré.

En aucun cas il ne saurait dispenser de la lecture d'un ouvrage sur UNIX.

Fondamentalement, pour pouvoir travailler sous UNIX vous avez besoin de savoir faire trois choses:

1. Éditer des fichiers texte. Pour cela on utilise un *éditeur*, il en existe un grand nombre (**`gedit`**, **`vi`** / **`gvim`**, **`emacs`**, ...).
2. Vous déplacer et manipuler l'arborescence des répertoires et des fichiers.
3. Exécuter des programmes.

Les deux dernières tâches se font le plus généralement dans un *interpréteur de commandes* ou *shell* (en anglais).

Note

Les IDE (ou *Integrated Developpement Environement*) sont, très schématiquement, des programmes qui fusionnent un éditeur de texte et l'exécution de commandes. Ils sont pratiques sous réserve de comprendre *quelles* commandes ils exécutent en sous-main et le cas échéant, comment les configurer. Nous n'aborderons pas leur cas dans ce court document.

1.1 Le Guide des Bonnes Manières

Les machines mises à votre disposition sont en réseau, y compris depuis l'extérieur. Cela signifie qu'un étudiant peut travailler *à distance* depuis chez lui sur une machine. Si la machine est redémarrée sans crier gare, cela peut entraîner la perte d'une partie des données.

Il est donc de bon ton de ne pas:

1. Redémarrer / arrêter une machine en marche.
2. Allumer une machine éteinte (il y a sûrement une bonne raison qu'elle le soit).
3. Ne pas oublier de quitter votre session avant de partir afin de ne pas verrouiller la machine.

2. Prélude: Se *Loguer* sur le Réseau

Avant de rentrer vos *login* et mot de passe, toujours vérifier que le verrouillage des majuscules est *désactivé* et que le pavé numérique *est* déverrouillé. De manière générale, n'utilisez jamais le pavé numérique pour saisir votre mot de passe.

3. Terminal, Interpréteur de Commandes et Environnement

Précisions sur la terminologie :

- *terminal* : le périphérique (au sens large) qui est utilisé pour lire les commandes saisies au clavier par l'utilisateur et en afficher le résultat. Les plus couramment utilisés en mode graphique sont **`xterm`**, **`gnome-terminal`** (sous Gnome) ou **`kterm`**; (sous KDE). En mode texte, la console *est* le terminal.
- *shell* : (*interpréteur de commandes*) le programme chargé de lire les commandes de l'utilisateur puis de les exécuter. Un *shell* interactif est exécuté dans un *terminal*. Le *shell* utilisé par défaut est le **`bash`** (ou *Bourne Again SHell*).

- *environnement* : c'est une collection de variables (dites d'*environnement*) permettant de configurer le comportement de nombreux programmes, en particulier du *shell* lui-même.

Dans la suite on fera l'abus de langage consistant à parler de *terminal* au lieu du *shell* s'exécutant dans le terminal.

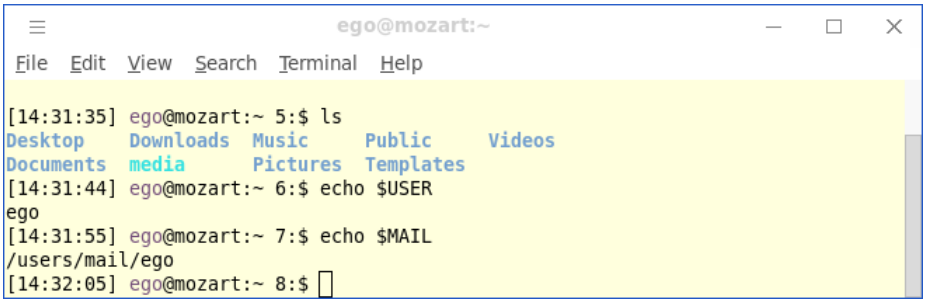

Figure 1: Exemple de terminal (Gnome/Xfce)

3.1 Manipulation de l'Environnement

Principales Variables d'Environement	
Nom	**Fonction**
`PATH`	Liste de répertoires séparés par `:` dans laquelle seront cherchés les programmes à exécuter. Les répertoires sont parcourus dans l'ordre jusqu'à trouver un fichier exécutable dont le nom correspond à celui de la commande, la recherche s'arrête sur le premier fichier trouvé. Exemple : `/usr/bin:/opt/arduino-1.6.8:/soc/alliance/bin`.
`MANPATH`	Liste de répertoires séparés par `:` dans laquelle seront cherchés les fichiers de l'aide en ligne (*man*). Exemple : `/usr/share/man:/usr/alliance/share/man`.
`MAIL`	Fichier dans lequel les *Mails* arrivant sont stockés. Cette variable doit valoir: `/var/spool/mail/$USER`. Noter que la plupart des lecteurs de *Mails* récents utilisent leurs propres fichiers de configuration pour positionner le chemin de ce fichier.

Positionner une variable d'environnement

```
ego@machine:~> export PATH=/usr/bin:/soc/alliance/bin
```

Note

Syntaxe: il ne doit y avoir aucun espace avant ou après le signe =.

Utiliser une variable d'environnement.

Pour faire référence à la valeur d'une variable d'environnement, faire précéder le nom de la variable du caractère `$`. Ici on utilise la variable **`USER`** qui vaut `ego`:

```
ego@machine:~> echo $USER
ego
ego@machine:~> export MAIL=/var/spool/mail/$USER
ego@machine:~> echo $MAIL
/var/spool/mail/ego
```

3.2 Transmission de l'Environnement

1. *Environnement initial (par défaut) :* au moment où vous vous connectez, les variables d'environnement sont initialisées par le système à une valeur par défaut raisonnable. Vous pouvez ajouter vos propres initialisations à l'environnement en éditant le fichier **`.bashrc`** dans la racine de votre compte.
2. *Héritage:* l'environnement qui est transmis aux processus fils est une copie de celui du père au moment où le fils est lancé. Conséquences :
   - Une fois le fils lancé, changer l'environnement du père n'affectera pas le fils.
   - Changer l'environnement du fils n'affecte pas le père.

Plus simplement: chaque terminal a son propre environnement. Si une commande ne s'exécute pas de façon identique dans deux terminaux différents, la première chose à vérifier c'est l'environnement. Les commandes **`env`** ou **`set`** affichent toutes les variables de l'environnement:

```
ego@machine:~> set
ALLIANCE_TOP=/dsk/l1/jpc/alliance/Linux.el7/install
DISPLAY=:0
EDITOR=vim
GID=XXXXX
HOME=/users/enseig/ego
HOST=machine
LANG=fr_FR.utf8
MAIL=/var/spool/mail/ego
MANPATH=/usr/share/man:/soc/alliance/share/man
PATH=/usr/bin:/soc/alliance/bin
PWD=/users/enseig/ego
UID=YYYYY
USER=ego
...
ego@machine:~>
```

3.3 Exécuter des Commandes

Pour exécuter un programme, il suffit de taper son nom suivi de ses éventuels arguments puis `RETURN`. Le *shell* va rechercher le programme dans la liste ordonnée des répertoires de la variable `PATH` puis le lancer. Il est aussi possible de donner un chemin relatif au répertoire courant ou bien complet du programme (et dans ce cas `PATH` n'est pas utilisé).

```
ego@machine:~> ls -a
.bashrc     .vimrc    Documents   Bureau
ego@machine:~> /usr/bin/ls -a
.bashrc     .vimrc    Documents   Bureau
ego@machine:~> ./MOBJ/TMEs/TME1/install/bin/tme1
Hello World!
ego@machine:~>
```

4. L'arborescence des répertoires et des fichiers

Les répertoires (le système de fichier) ont une structure arborescente où la racine (ou *root*, `/`) se trouve «en haut».

Sur le réseau enseignement, la racine du compte d'un utilisateur dont le **`login`** est `ego` sera `/users/enseig/ego`.

Note

Explorateur de Fichiers: il ne permet pas de réaliser autant d'opérations qu'un *shell* peut et parfois entretenir une certaine confusion sur le répertoire dont le contenu est affiché. Vous êtes donc *fortement* encouragé à acquérir les notions de navigation dans l'arborescence à partir d'un *shell*.

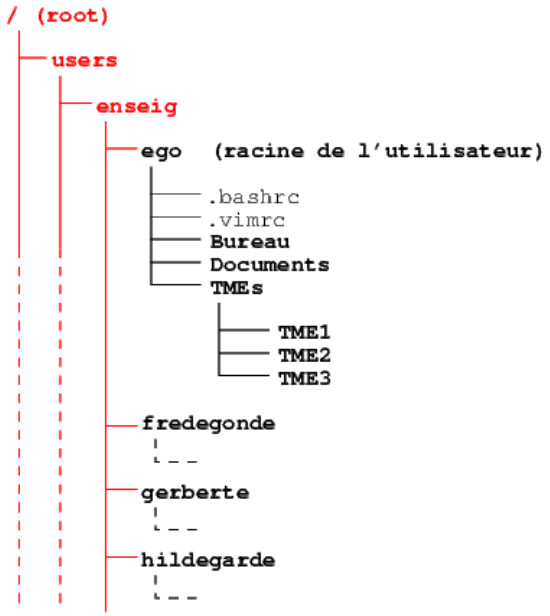

Figure 2: Arborescence du système de fichiers

Il existe un certain nombre d'abréviations reconnues par le *shell* pour désigner les répertoires:

Notation	Signification
~	La racine du compte de l'utilisateur, par exemple `/users/enseig/ego`.
.	Le répertoire courant
..	Le père du répertoire courant

4.1 Où suis-je ?

Pour connaître le chemin complet du répertoire courant depuis la racine: **`pwd`**

```
ego@mozart:TMEs> pwd
/users/enseig/ego/TMEs
ego@mozart:TMEs>
```

4.2 Voir le contenu d'un répertoire

Les commande **`ls`** et **`ll`** vous renseignent

```
ego@mozart:TMEs> ls
TP1/  TP2/  TP3/
ego@mozart:TMEs> ll
drwxr-xr-x    2 ego    M1       512 Sep 24 12:11 TME1/
drwxr-xr-x    2 ego    M1       512 Sep 24 12:11 TME2/
drwxr-xr-x    2 ego    M1       512 Sep 24 12:11 TME3/
ego@mozart:TMEs>
```

4.3 Se déplacer dans l'arborescence

La commande **cd**. **cd** sans arguments repositionne dans le répertoire racine de l'utilisateur (identique à `cd ~`, mais plus court).

```
ego@mozart:TMEs> pwd
/users/enseig/ego/TMEs
ego@mozart:TMEs> cd TME1
ego@mozart:TME1> pwd
/users/enseig/ego/TMEs/TME1
ego@mozart:TME1> cd ../TME2
ego@mozart:TME2> pwd
/users/enseig/ego/TMEs/TME2
ego@mozart:TMEs> cd ..
ego@mozart:TMEs> pwd
/users/enseig/ego/TMEs
ego@mozart:TMEs> cd ~
ego@mozart:ego> pwd
/users/enseig/ego
ego@mozart:ego>
```

4.4 Création et destruction des répertoires

mkdir crée un nouveau répertoire, **rmdir** l'efface (**rmdir** ne supprime que les répertoires vides).

```
ego@mozart:TMEs> pwd
/users/enseig/ego/TMEs
ego@mozart:TMEs> mkdir TME4
ego@mozart:TMEs> ls
TME1/  TME2/  TME3/  TME4/
ego@mozart:TMEs> rmdir TME4
ego@mozart:TMEs> ls
TME1/  TME2/  TME3/
ego@mozart:TMEs>
```

4.5 `rm` ou *Vade Retro Satanas*

La commande **rm** est très dangereuse, elle doit toujours être utilisée avec précaution. Elle permet d'effacer des fichiers ou des arborescences entières récursivement (option `-r`). Une option particulièrement dangereuse est `-f`: elle *force* (supprime toute forme de confirmation) avant effacement. Combinée à l'option `-r` elle est donc particulièrement redoutable.

4.6 Les Aventuriers du Fichier Perdu: `find`

Il arrive (mais c'est très rare) que l'on ne sache plus où l'on a rangé un fichier, ou bien qu'un autre utilisateur nous ait simplement donné un nom...

Pour remettre la main dessus vite fait la commande **find** est là! On lui fournit comme premier argument la racine de l'arborescence à explorer puis le *pattern* du nom de fichier à rechercher: `-name "monFichier"` (ne pas oublier les guillemets autour du *pattern*).

find est capable de recherches beaucoup plus complexes, `man find` pour tout savoir.

Note

Ne pas lancer de recherche sur la racine du système de fichier, cela peut ralentir considérablement le serveur de fichier.

Recherche des fichiers **pdf** dans un compte:

```
ego@mozart:~> cd
ego@mozart:~> find . -name "*.pdf"
./DesignKit_SXBUILD/doc/CDS/LEFDEF-preface.2.pdf
./Steiner/book.pdf
./Steiner/18.pdf
./Steiner/AlgsInRealWorld.pdf
./page.pdf
ego@mozart:~>
```

5 Gestion des Programmes / *Processus*

Encore quelques définitions:

- Un *Processus* est une instance d'un programme en train de s'exécuter sur une machine (il peut y avoir plusieurs *Processus* d'un même programme). Dans la suite on préfèrera le terme de *Processus* à celui de programme.
- Pour identifier chaque *Processus* on lui associe un numéro unique, son `pid` (ou *Processus IDentifier*).
- Un *Signal* est une interruption (logicielle ou matérielle) envoyée par le système d'exploitation ou l'utilisateur à un *Processus*. Les plus utilisés sont:
  - `sigterm` : demande à un *Processus* de s'arrêter proprement.
  - `sigkill` : provoque l'arrêt immédiat et inconditionnel d'un *Processus*.

5.1 Quels *Processus* s'exécutent actuellement?

La commande **ps** est là pour ça. Pour «voir» tous les *Processus*, utiliser les arguments `aux`.

```
ego@mozart:~> ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.2   1388   524 ?        S    Sep19   0:05 init [5]
root       431  0.0  0.2   1452   596 ?        S    Sep19   0:01 syslogd -m 0
root       436  0.0  0.4   2040  1140 ?        S    Sep19   0:00 klogd -2
rpc        450  0.0  0.2   1572   692 ?        S    Sep19   0:00 portmap
rpcuser    465  0.0  0.2   1588   760 ?        S    Sep19   0:00 rpc.statd
root       553  0.0  0.7   2004  1996 ?        SL   Sep19   0:05 ntpd
root       629  0.0  0.3   5736   780 ?        S    Sep19   0:00 ypbind
root       785  0.0  0.4   2632  1132 ?        S    Sep19   0:03 /usr/sbin/sshd
root       805  0.0  0.3   2300   996 ?        S    Sep19   0:00 xinetd -stayalive
root      7201  0.0  0.7   3448  1808 ?        S    Sep23   0:00 /usr/sbin/sshd
root      7980  0.0  0.8   6408  2088 ?        S    Sep23   0:00 /usr/bin/gdm -nod
root      7981  0.0  3.1  60816  8108 ?        S    Sep23   0:01 /etc/X11/X :0 vt7
gdm       7986  0.0  1.5   7356  3864 ?        S    Sep23   0:00 /usr/bin/gdmlogin
ego      12358  0.1  0.5   2496  1392 pts/0    S    09:26   0:00 -bash
ego      12431  0.0  0.3   2832   900 pts/0    R    09:28   0:00 ps aux
ego@mozart:~>
```

5.2 Tuer un *Processus*

La commande **kill** vous permet d'envoyer un signal à un *Processus*. Bien entendu, vous ne pouvez atteindre que les *Processus* qui *vous* appartiennent.

Note

Kill ne doit être utilisée qu'en dernier recours normalement, au moment de votre déconnexion, tous les *Processus* vous appartenant sont supprimés. Mais il peut arriver qu'un programme se soit bloqué, ou que vous ayez dû l'interrompre par un `CTRL-C`, dans ce cas, s'il persiste en mémoire il faut le tuer avec **kill**.

Exemple:

```
ego@mozart:~> kill -KILL 12358
ego@mozart:~>
```

6. Autres Opérations de Base

6.1 Au secours: où trouver de l'aide!

La première source d'information sous UNIX est le manuel en ligne. On l'invoque avec la commande **man** et en argument la commande ou le mot pour lequel on cherche de l'aide.

La page de manuel de `rmdir`:

```
ego@machine:~> man rmdir
RMDIR(1)                        FSF                        RMDIR(1)

NAME
       rmdir - remove empty directories

SYNOPSIS
       rmdir [OPTION]... DIRECTORY...

DESCRIPTION
       Remove the DIRECTORY(ies), if they are empty.

       --ignore-fail-on-non-empty

              ignore each failure that is solely because a direc
              tory is non-empty

       -p, --parents
              remove DIRECTORY, then try to remove each directory
              component  of  that  path  name.   E.g.,  'rmdir -p
              a/b/c' is similar to 'rmdir a/b/c a/b a'.

       -v, --verbose
              output a diagnostic for every directory processed

       --help display this help and exit

       --version
              output version information and exit

AUTHOR
       Written by David MacKenzie.

REPORTING BUGS
       Report bugs to <bug-fileutils@gnu.org>.

COPYRIGHT
       Copyright © 2000 Free Software Foundation, Inc.
       This is free software; see the source for  copying  condi
       tions.  There is NO warranty; not even for MERCHANTABILITY
       or FITNESS FOR A PARTICULAR PURPOSE.

SEE ALSO
       The full documentation for rmdir is maintained as  a  Tex
       info  manual.  If the info and rmdir programs are properly
       installed at your site, the command
```

```
                info rmdir

        should give you access to the complete manual.

GNU fileutils 4.0.36       January 2001                 RMDIR(1)
ego@machine:~>
```

6.2 Se connecter à distance : `ssh`

Cette opération permet d'exécuter des commandes sur une machine autre que celle devant laquelle on se trouve physiquement. En particulier, c'est la seule façon de travailler sur les serveurs de calcul qui se trouvent en chambre froide.

Une session **`ssh`** commence par l'appel à la commande **`ssh`** avec pour argument le nom de la machine sur laquelle on veut se connecter, ici `berlioz`. Pour terminer une session, exécuter la commande **`logout`** ou taper la combinaison de touches `CTRL-D`.

```
ego@mozart:~> ssh berlioz
The authenticity of host 'berlioz (132.227.67.141)' can't be established.
RSA key fingerprint is 5d:12:5f:c5:1e:fc:a7:ec:af:88:99:73:af:64:6a:4f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'berlioz,132.227.67.141' (RSA) to the list of known hosts.
Last login: Sun Sep 25 15:52:41 2016 from mozart

Une petite documentation sur le réseau enseignement est disponible ici:
        https://soc-extras.lip6.fr/fr/enseignement/enseig/

ego@berlioz:~> # On est sur <berlioz>.
ego@berlioz:~> logout
ego@mozart:~>
```

Note

Avant de terminer une session ssh assurez vous que tous les programmes que vous avez lancés sur la machine distante sont bien terminés. Si vous l'oubliez, certains d'entre eux peuvent continuer à dévorer le CPU ad infinitum, pénalisant tous les utilisateurs...

Note

Programmes graphiques pour les connaisseurs d'X-Window, **`ssh`** positionne correctement la variable `DISPLAY` sur la machine distante de façon à ce que les applications graphique s'affichent sur la machine d'origine (en utilisant un *tunnel*). Il ne faut donc pas essayer de la repositionner.
Si les applications graphiques ne s'affichent pas depuis la machine distante sur la machine locale, vous pouvez forcer la redirection en ajoutant les les *flags* `-X` **ou** `-Y`:

```
ego@mozart:~> ssh -X berlioz
ego@berlioz:~>
```

6.3 Connaître son `quota`

L'espace disque mis à votre disposition est limité. Initialement les comptes de Master 1 sont limités à 1Go et les comptes de Master 2 à 2Go. Cette limite peut évoluer en fonction des besoins spécifiques des stages (requête à effectuer par l'encadrant de stage).

Pour connaître l'état d'un compte, utiliser la commande `quota -v` qui retourne le nombre de blocs de 1Ko utilisés.

```
ego@home:~> quota -v
Disk quotas for user ego (uid 10XXX):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
disco-enseig:/users/disco4/enseig
                   516 1024000 1024000             274   51200   51200
ego@home:~> du -ks .gnome2
87K     .gnome2
ego@home:~>
```

Dans cet exemple l'utilisateur `ego` utilise 516 blocs de 1Ko (dans la colonne **blocks**) sur un total disponible de 1024000Ko (ou 1Go, dans la colonne **quota**). Pour connaître la taille d'une arborescence seule, il existe la commande **`du`**. Dans l'exemple ci-dessus le répertoire `.gnome2` occupe 87Ko.

6.4 Jolis fichiers textes: `a2ps`

Pour obtenir une belle version imprimable d'un fichier texte, passez par **`a2ps`**. **`a2ps`** met en page et fait une colorisation syntaxique pour un grand nombre de formats (en particulier le C, le C++ et le VHDL). Vous obtiendrez des fichiers (beaucoup) plus lisibles et vous utiliserez moité moins de papier (l'Amazonie vous le rendra).

```
ego@mozart:~> a2ps Box.h -o Box.h.ps
```

7. Imprimer : `lpr`

Une imprimante, **`sibelius`**, est disponible en salle 305, elle est configurée pour être l'imprimante par défaut.

Note

Quota d'impression: chaque utilisateur dispose d'un quota de 100 pages par semaine. Au dela de cette limite vos impressions seront automatiquement refusées. L'imprimante est recto-verso, pensez à utiliser cette fonctionalité.

La plupart des logiciels d'édition de fichiers disposent de boîtes de dialogue permettant d'imprimer directement.

Dans un terminal, vous disposez de trois commandes:

- **`lpr`** pour envoyer un fichier vers la file d'attente de l'imprimante.
- **`lpq`** pour voir la liste des fichiers dans la file d'attente (et leur possesseurs).
- **`lprm`** pour effacer un de *vos* fichiers de la file d'attente.

Note

Avec **`lpr`**, n'envoyez à l'impression que des fichiers de type *texte*, *ps* ou *pdf*.

Exemple:

```
ego@mozart:~> lpr mon_beau_CV.pdf
ego@mozart:~> lpq
sibelius is ready and printing
Rank    Owner   Job     File(s)                         Total Size
active  remroot 2459    mon_beau_CV.ps                  1024 Kb
1st     tintin  2460    milou.pdf                       2048 bytes
ego@mozart:~> lprm 2459
```

8. Éditer du Texte

Comme dit en introduction, il existe une très grande variété d'éditeurs de texte. En fonction de votre niveau de compétence, vous pouvez utiliser:

1. **`gedit`**, le plus simple à prendre en main, mais ses fonctionnalités et son ergonomie sont limitées.
2. **`vi`** (dans un terminal) ou **`gvim`** (en mode graphique), très puissant mais d'un abord déconcertant en raison de la séparation des modes *INSERT* et *COMMAND*.
3. **`emacs`** lui aussi très efficace et disposant d'une très large bibliothèque d'extensions. Nécessite néanmoins un temps d'apprentissage.

8.1 `gedit`

Figure 3: Éditeur de texte minimal: `gedit`

8.2 Rester Zen sous `vi`

`vi` est un éditeur de texte extrêmement puissant, mais déconcertant au premier abord. Sa prise en main délicate découle du fait qu'il dispose de deux modes de gestion du clavier:

- Un mode *INSERT* : quant vous tapez `azertyuiop`, `azertyuiop` est inséré dans votre texte (à l'endroit du curseur).
- Un mode *COMMAND* : le clavier (TOUTES les touches) servent à composer des ordres pour **`vi`**. Par exemple `:w` sauvegardera votre texte sur disque.

Note

Ne pas taper frénétiquement sur le clavier si vous ne savez plus dans quel mode vous êtes. Il y a de fortes chances que vous soyez passés en mode *COMMAND* et que vous donniez des ordres loufoques, plus vite vous arrêterez, moindres seront les dégâts...

8.2.1 Comment distinguer les modes *INSERT* et *COMMAND*

La dernière ligne de votre terminal n'affiche pas le texte en cours d'édition mais l'état de **`vi`** : c'est la ligne d'état.

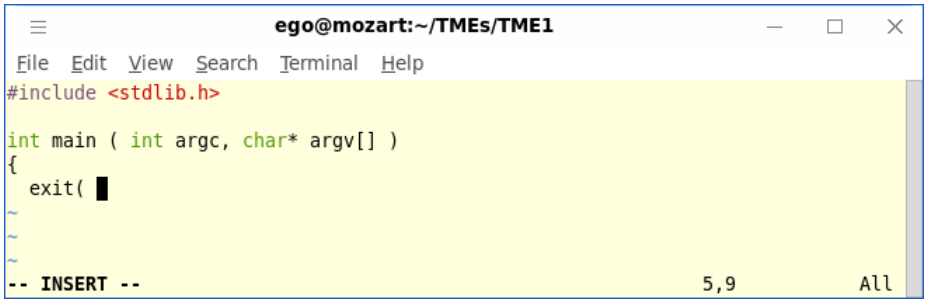

Figure 4: `vi` en mode *INSERT*

`vi` est en mode insertion *si, et seulement si* au début de la ligne d'état est affiché:

```
-- INSERT --
```

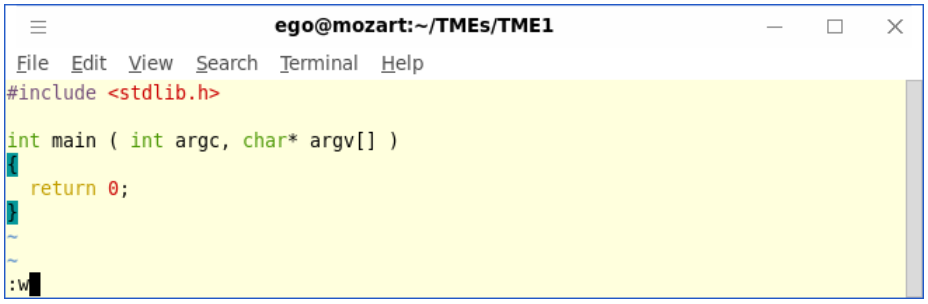

Figure 5: `vi` en mode *COMMAND*

Dans tous les autres cas, vous êtes en mode *COMMAND* (Ici, sur le point d'éxécuter la commande `:w` pour sauvegarder un fichier).

8.2.2 Passages entre les modes *INSERT* et *COMMAND*

Commandes provoquant le passage en mode *INSERT* : vous tapez cette commande *alors* que vous êtes en mode *COMMAND* et vous passez immédiatement en mode *INSERT*.

Commandes de passage en mode insertion	
Commande	Type d'insertion
`i` (*insert*)	Insère *avant* le curseur
`I` (*insert*)	Insère *au début* de la ligne courante
`a` (*append*)	Insère *après* le curseur
`A` (*append*)	Insère *à la fin* de la ligne courante
`o`	Insère *sur une nouvelle ligne* ajoutée en dessous de la ligne courante
INSERT et remplacement	
`cw` (*Change Word*)	Remplace le mot sous le curseur

Retour au mode *COMMAND* : tapper `ESC` (`Echap` sur un clavier français) ou `CTRL-C`.

8.2.3 Commandes de base

Effacement & sauvegarde	
Commande	**Action**
`x`	Efface le caractère *sous* le curseur
`dw` (*delete word*)	Efface le mot sous le curseur
`dd`	Efface la ligne courante
`u` (*undo*)	Annule la dernière commande
`.`	Exécute à nouveau la commande précédente
`:w`	Sauvegarde le fichier
`:q`	Quitte **`vi`**
`:q!`	Quitte en annulant tous les changements effectués depuis le dernier `:w`

Déplacements dans un fichier

Commande	**Mouvement**
`h`	Un caractère à gauche
`j`	une ligne vers le bas (même colonne)
`k`	un caractère à droite
`i`	une ligne vers le haut (même colonne)
`|`	va en début de ligne
`$`	va en fin de ligne
`:10`	va à la dixième ligne
`:$`	va à la dernière ligne
`w`	va au début du mot suivant
`b`	va au début du mot précédent
`e`	va à la fin du mot suivant

Déplacement de blocs de texte

Commande	Action
Copie vers le *buffer*	
`yy`	Copie la ligne courante
`12yy`	Copie 12 lignes (à partir de la ligne courante)
`11yl`	Copie 12 caractères (à partir du curseur)
Coupe vers le *buffer*	
`dd`	Coupe la ligne courante
`12dd`	Coupe 12 lignes (à partir de la ligne courante)
`11x`	Coupe 11 caractères (à partir du curseur)
Collage du contenu du *buffer*	
`p`	Colle le contenu du buffer après la position courante du curseur
Autres	
`:r <filename>`	Insère le contenu du fichier `filename` après la ligne courante

Recherche

Commande	Action
`/<PATTERN>`	Recherche *en avant* de `<PATTERN>`
`?<PATTERN>`	Recherche *en arrière* de `<PATTERN>`
`n`	Recherche de l'occurrence suivante du `<PATTERN>` dans la direction courante (avant ou arrière)

Recherche et substitution

Commande	Action
`:%s/<PATTERN>/<REPLACE>/`	Recherche `<PATTERN>` dans la totalité du fichier et remplace chacune des occurrence par `<REPLACE>`