
INTRODUCTION A L'EDITEUR ed

Ed est l'éditeur standard de Unix. Unix supporte également d'autres éditeurs, tels que emacs de J. Gosling, edith de P. Amar, etc, mais ed est le plus simple à apprendre et marche sur tout terminal. C'est un éditeur de lignes, c'est-à-dire : toute commande se réfère à des lignes de fichiers.

Pour lancer l'éditeur, taper

```
ed
```

ou

```
ed nom-de-fichier
```

Si le fichier existe déjà, ed répond par le nombre de caractères du fichier, sinon ed répond par

```
?nom-de-fichier
```

indiquant par cela que le fichier du nom donné n'existe pas encore. Dans les deux cas, ed est prêt à lire et à exécuter vos commandes.

Les Commandes De Base

Supposons que nous voulions créer un nouveau fichier de nom fred :

```
$ ed fred
?fred
```

Naturellement, la première chose à faire est d'y écrire quelque chose. Ajouter quelque chose à un fichier se fait avec la commande a (pour ajouter).

La séquence

```
a
au village, sans prétention
j'ai mauvaise réputation
que je me démène, ou qu'je reste coi
je passe pour un je ne sais quoi
```

ajoute le texte entre le a de la commande et le point. ed ne donne aucun prompt-caractère et ne fait aucun contrôle sur les caractères que vous entrez. Jusqu'à l'instant où vous donnez une ligne ne contenant qu'un point en première position, tout ce que vous tapez sera pris comme du texte.

Une fois que vous avez terminé la frappe, il faut écrire ce texte dans le fichier fred. Ceci est fait par la commande w (pour write).

```
w
126
q
$
```

ed répond à la commande w en vous donnant le nombre de caractères écrits. La commande q (pour quit) vous sort ensuite de l'éditeur et vous vous retrouvez sous shell.

Vous pouvez afficher le contenu du fichier pour vérifier que tout marche bien, avec la commande shell cat.

Maintenant que ce fichier fred existe, nous pouvons l'éditer à nouveau :

```
$ ed fred
126
```

Comme toujours, ed nous répond avec le nombre de caractères du fichier et attend une commande.

A tout instant vous pouvez écrire dans le fichier fred votre version actuelle. Il est particulièrement recommandé de faire cela de temps à autre quand vous éditez un grand fichier : tant que vous n'avez pas donné un ordre d'écriture w, tous vos changements, rajouts, etc, ne sont qu'à l'intérieur d'un buffer de l'éditeur et le fichier édité n'est pas modifié. Imaginez le désastre, si après cinq heures d'édition la machine tombe en panne et vous n'avez pas une seule fois sauvegardé (écrit sur le fichier) vos travaux !

Si vous voulez écrire sur un autre fichier, donnez la commande w avec, comme argument, le nom de cet autre fichier.

Pour imprimer (voir!) une ligne, donnez la commande p (pour print). Si vous donnez cette commande maintenant, ed répond par la dernière ligne :

```
je passe pour un je ne sais quoi
```

ed a un curseur qui, après le lancement, pointe vers la dernière ligne du fichier. Ainsi, si vous donnez maintenant la commande a, le texte suivant cette commande sera bien rajouté à la fin du fichier.

Vous pouvez donner un argument à p, pour ne pas imprimer que la dernière ligne. Ainsi

2p

imprime la deuxième ligne, et

1p

imprime la première. Vous pouvez également donner un interval de lignes :

1,3p

imprimera les lignes 1 à 3, et

1,\$p

imprimera le fichier en entier. Le signe \$ réfère, en ed, à la dernière ligne du fichier.

En fait, la plupart des commandes de ed admettent des numéros de lignes les précédents. Ainsi la commande s'applique à tout l'intervalle (ou juste à la ligne si l'on ne donne qu'un numéro de ligne) donné.

La commande r lit un fichier et la commande d (pour delete) efface une ligne. Voici quelques exemples :

1p imprime la ligne 1

1,4p imprime les lignes 1 à 4

5r fred lit le fichier fred et positionne le texte après la ligne 5

1,6w temp écrit les lignes 1 à 6 dans le fichier temp

2d efface la ligne 2

1,2d efface les premières 2 lignes

3a ajoute le texte suivant après la ligne 3

Evidemment, les numéros de lignes précédents une commande permettent de déplacer le curseur de ed.

Ainsi, vous pouvez toujours parler d'une ligne courante : c'est sur cette ligne qu'aura lieu votre prochaine commande.

La ligne courante est dénotée par un "." (un point) et la dernière ligne est dénotée par un "\$" (dollar). Les numéros de lignes précédant les commandes positionnent le point, ils modifient ce qui est considéré comme la ligne courante. Ainsi la commande

5p

n'imprime pas seulement la cinquième ligne, mais, également, elle place le point dans la cinquième ligne.

Afin de savoir dans quelle ligne vous êtes, la commande

.=

vous donne le numéro de la ligne courante.

Si vous donnez juste un nombre n, ed suppose que vous voulez vous positionner sur la ligne numéro n et l'imprimer. Ainsi

4 vous positionne sur la ligne numéro 4 et l'imprime

.-1 vous positionne sur la ligne précédente et l'imprime

.-6, .+ 6 imprime les 6 lignes précédentes et les 6 lignes suivantes et vous positionne 6 lignes plus loin (en ligne ".+ 6").

Pour juste avancer d'une ligne, un return est suffisant et pour reculer de juste une ligne, la commande "-" (moins) fait l'affaire.

Une des commandes les plus importantes est la commande de substitution s. Elle est utilisée pour remplacer une partie de texte par une autre.

Reprenons notre exemple et positionnons-nous dans la première ligne qui est

au village, sans prétention

et essayons :

s/san/sans/p
au village, sanss prétention

Le mot san a été changé en sans. Le p à la fin ne sert qu'à imprimer la ligne modifiée; la commande de substitution seule n'imprime rien.

La forme générale de cette commande est :

ligne_i, ligne_j s/pour ceci/cela/

qui remplace la première occurrence de `pour ceci` dans les lignes `lignei` à `lignej` en `cela`. Pour remplacer toutes les occurrences dans un intervalle de lignes donné, il faut donner la commande

`lignei, lignej s/pour ceci/cela/g`

où la lettre `g` est un ordre de `ed` indiquant d'appliquer globalement la commande.

Pour enlever une partie de texte, on peut la remplacer par rien. Ainsi

`s/xyz//`

transforme la première occurrence de `xyz` dans la ligne courante en rien.

N'utilisez pas, pour l'instant, les caractères `^`, `.`, `$`, `[`, `*`, `\` et `&` dans vos commandes. Ils ont une signification particulière.

La commande

`/abc/`

cherche la première occurrence de la séquence `abc` et positionne le point sur cette ligne. Pour faire une recherche vers l'arrière, il faut donner la commande :

`?abc?`

qui cherche la première occurrence de la séquence avant la ligne courante.

`/qqc/` et `?qqc?` sont interprétés par `ed` comme des numéros de lignes. Par exemple, la commande

`/a/,b/s/234/456/g`

cherche la première ligne après la ligne courante qui contient un caractère `a`. Sur cette ligne et sur toutes les lignes suivantes jusqu'à ce qu'on tombe sur une ligne contenant un caractère `b`, toutes les occurrences de la séquence `234` sont changés en `456`.

`ed` se rappelle de la dernière recherche et si vous cherchez la troisième occurrence d'un certain filtre, pour la deuxième et troisième recherche, il suffit de donner la commande `//`.

Par exemple, si vous cherchez la quatrième occurrence de la chaîne `soleil`, la suite des commandes suivantes :

`/soleil/
//`

```
//
//
```

vous positionne sur la ligne contenant cette quatrième occurrence. Ce filtre mémorisé peut être également utilisé dans des commandes de substitution.

```
/Deadeye Dick/s/Mexico Pete/
//s//
```

cherche une ligne contenant Deadeye Dick, si ed en trouve une, cette occurrence est changée en Mexico Pete. Ensuite, ed continue à chercher une deuxième occurrence et si la recherche a du succès, la chaîne Deadeye Dick est enlevée.

Voici une dernière commande, la commande m, de forme générale

ligne_i, ligne_j m après-cette-ligne

qui prends les lignes de textes comprise entre ligne_i et ligne_j et les place après la ligne après-cette-ligne.

Exemple :

7,12m\$

place les lignes 7 à 12 à la fin du fichier.

Les Caractères Spéciaux

Les caractères ^, ., \$, [, *, \ et & ont une signification particulière. Les cinq premiers caractères sont utilisés pour le filtrage (similaires aux métacaractères de shell).

Le caractère “\” est le quote caractère : il permet d’utiliser les autres caractères spéciaux comme des caractères normaux. Ainsi

```
s/^\*\&/abc/
```

change la chaîne *\& en la chaîne abc.

^ Ce caractère filtre le 0^{ème} caractère d’une ligne. Il sert à trouver des mots au début de la ligne.

```
/^ premier/
```

cherche la première occurrence de la chaîne premier au début d’une ligne, et

```
s/^ /ok/
```

insère la chaîne ok au début de la ligne.

\$ en tant que numéro de ligne, le dollar (\$) désigne la dernière ligne, en tant que filtre, il désigne la fin d'une ligne :

`s/$/fin/`

insère la chaîne fin en fin de ligne, et

`/^ juste ceci$/`

cherche une ligne qui ne contient que la chaîne juste ceci.

. Le point, quand il est utilisé comme numéro de ligne, désigne la ligne courante, à l'intérieur d'un filtre, il représente un caractère quelconque non spécifié.

`/a.b/`

filtre :

aab
abb
a.b
acb
etc

et

`/^ ...$/`

cherche une ligne d'exactly trois caractères.

* L'étoile indique un nombre quelconque du même caractère que celui qui précède l'étoile.

`/x*/`

se lit : un nombre quelconque de caractères x, et

`/abc*/`

filtre

ab
abc
abcc

