Chapitre **5**

Éditeurs de texte

L'éditeur de texte de base de Linux est un programme appelé `ed`, écrit par Ken Thompson. `ed` a été conçu vers 1970 pour fonctionner dans un environnement de petites machines. Malgré les évolutions technologiques, `ed` n'a pas changé et il est aujourd'hui toujours disponible sur tous les systèmes Linux.

Sur Linux, on dispose bien sûr de l'éditeur `ed`, mais aussi d'un autre **éditeur ligne** `ex` et d'un **éditeur plein écran** `vi`. L'éditeur `ex` possède les mêmes commandes que l'éditeur `ed` à quelques améliorations près ; notamment les messages d'erreur sont donnés explicitement au lieu d'un simple ? sur `ed`.

Il existe d'autres éditeurs, en mode caractères (`emacs`,..) et en mode graphique X11 (`nedit`,...). Nous ne présentons ici que l'éditeur `vi` et un sous-ensemble des commandes de `ex` (recherche, substitution, copie, etc.). En effet, il est intéressant de connaître cet éditeur car il est disponible sur tous les systèmes Linux. Cet éditeur nous permettra, par exemple, l'écriture rapide de fichiers de commandes (voir chapitre 8).

Sous Linux, la commande `vi` lance l'éditeur `vim` correspondant à une version améliorée de `vi`.

L'éditeur `vi`, fonctionnant en pleine page, est essentiellement utilisé pour la création de nouveaux fichiers ou pour des modifications ponctuelles. `ex` est par contre plus adapté pour les recherches de motif ou pour des modifications systématiques. Comme sous `vi`, il est possible d'exécuter des commandes de `ex` en les faisant précéder d'un « : », ce sont les commandes de `ex` que nous utiliserons pour effectuer un traitement global (liste et effacement sélectif, substitution), les mouvements ou copies de groupes quelconques de lignes, la sauvegarde du fichier en cours d'édition ou la sortie de `vi`.

La figure 5.1 indique les cinq modes de fonctionnement de l'éditeur `vi` et les transitions possibles entre ces modes. Après l'appel de l'éditeur, `vi` est en mode

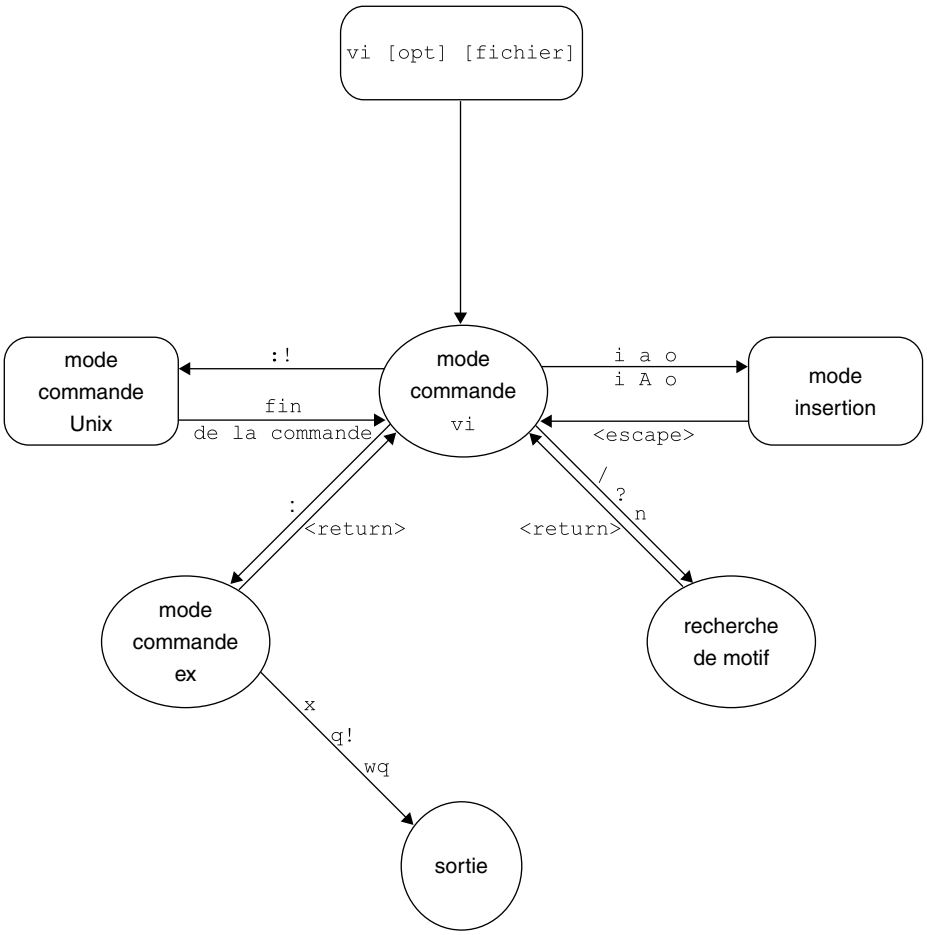


FIGURE 5.1. REPRÉSENTATION SYNOPTIQUE DU FONCTIONNEMENT DE L'ÉDITEUR *VI*.

commande, c'est-à-dire en attente d'une commande. Depuis ce mode, il est possible de passer :

- en mode insertion, le retour au mode commande étant réalisé à l'aide de la touche <Esc>,
- en mode recherche de motif,
- en mode commande de l'éditeur *ex*,
- en mode exécution d'une commande Linux.

Nous verrons que la plupart des commandes de *vi* correspondent à des touches du clavier ou à des combinaisons de touches. Cet éditeur fonctionne sur tout système

Linux du fait que *vi* a connaissance du type de terminal sur lequel vous travaillez grâce à la variable d'environnement TERM (voir chapitre 6). Cette variable indique à *vi* les caractéristiques du terminal.

Remarque

Si la variable d'environnement TERM n'a pas été positionnée, à l'appel de *vi*, un message d'erreur vous indiquera que l'éditeur ne reconnaît pas le type de terminal (voir chapitre 6).

5.1 L'ÉDITEUR PLEINE PAGE VI

5.1.1 Appel de l'éditeur et sorties

`vi nom de fichier`	appel de l'éditeur ; le curseur se positionne au début du fichier,
`vi r nom fichier`	idem mais pour récupérer un fichier après un problème machine,
`vi R nom fichier`	appel de l'éditeur mais avec verrouillage du fichier en lecture seule,
`ZZ (ou :x)`	mise à jour du fichier et sortie de l'éditeur,
`:wq`	idem à ZZ,
`:q!`	sortie de l'éditeur sans modification du fichier.

5.1.2 Renseignements utiles

`:set ts`	affiche le nombre de caractères d'une tabulation,
`:set ts=n`	fixe la tabulation à n caractères (par défaut n=8),
`:set nu`	affiche les numéros de lignes,
`:set nonu`	supprime les numéros de lignes,
`:.=`	affiche le numéro de la ligne courante,
`<ctrl g> (ou :f)`	affiche le nom et l'état du fichier.
`:e!`	Réédite le fichier tel qu'il est sur le disque

5.1.3 Déplacements de la page affichée

`<ctrl b>`	page précédente (back),
`<ctrl f>`	page suivante (forward).

5.1.4 Déplacements du curseur

La saisie d'un des caractères suivant nous amène dans la position indiquée.

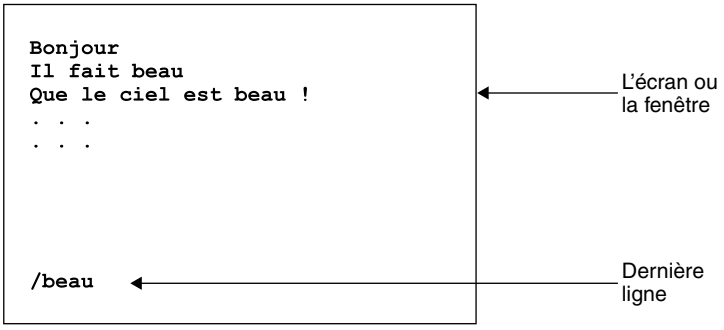
`G`	Dernière ligne,
`nG`	Ligne n,
`H`	Haut de l'écran (high),
`M`	Milieu de l'écran (middle),
`L`	Bas de l'écran (low),
`n<espace>`	n caractères vers la droite,
`0 (zéro)`	Début de ligne,
`^`	Premier caractère non blanc,
`$`	Dernier caractère de la ligne,
`↑ (ou k)`	Ligne précédente,
`← (ou h)`	Vers la gauche,
`→ (ou l)`	Un caractère vers la droite,
`↓ (ou j)`	Ligne suivante,
`n<return>`	n lignes vers le bas,
`+ ou <return>`	Début de la ligne suivante,
	Début de la ligne précédente.

5.1.5 Recherche

`/motif<return>`	recherche motif vers l'avant,
`?motif<return>`	recherche motif vers l'arrière,
`n`	répète la dernière commande de recherche.

Exemple

Voici un texte en vi :



La dernière ligne visible à l'écran en *vi* est une ligne de commandes. Lors de la mise à jour du document, la ligne « /beau » n'est pas sauvegardée.

5.1.6 Insertion

Le mode insertion est obtenue en tapant au choix I, i, a, A, o, O suivi du texte à insérer et terminé par la touche <Esc>. Cette touche remet l'éditeur en mode commande (en cas de doute, taper encore une fois <Esc> ; le signal sonore fera savoir que l'on n'est plus en mode insertion). Les commandes d'insertion sont :

`i… <Esc>`	avant le curseur,
`I… <Esc>`	en début de ligne courante,
`a… <Esc>`	après le curseur,
`A… <Esc>`	en fin de ligne courante,
`o… <Esc>`	au-dessous de la ligne courante,
`O… <Esc>`	au-dessus de la ligne courante.

5.1.7 Caractères spéciaux en mode insertion

`<del> ou <backspace>`	efface le dernier caractère inséré,
`<ctrl u>`	annule le texte inséré de la ligne courante,
`<ctrl v>`	permet d'insérer le caractère spécial qui suit,

5.1.8 Remplacement

`r#`	remplace le caractère courant par le caractère #,
`R… <Esc>`	remplace la chaîne de caractères courante par les caractères compris entre R et <Esc>,
`~`	bascule de MAJUSCULE en minuscule et inversement,
`.`	répète la dernière commande modifiant un texte.

5.1.9 Effacement

`x`	du caractère sous le curseur,
`3x`	de trois caractères à partir de la position du curseur,
`dd`	de la ligne où se trouve le curseur,
`4dd`	de quatre lignes à partir de la ligne où est le curseur.

5.1.10 Restitution

`u`	dernière modification (*undo*),
`U`	la ligne courante avant modification (sauf *dd*),
`p`	dernière commande d'effacement.

5.1.11 Mouvements de lignes

`J`	joint la ligne courante et la suivante,
`r<return>`	brise en deux lignes à partir de la position du curseur,
`yy`	place la ligne courante dans le tampon de travail,
`dd`	idem avec effacement de la ligne,
`10yy`	place dix lignes dans le tampon de travail,
`10dd`	idem avec effacement des dix lignes,
`p`	insère le contenu du tampon sous la ligne courante,
`"ayy`	place la ligne courante dans le tampon a (un tampon nommé se conserve au changement de fichier),
`"anyy`	place n lignes dans le tampon a,
`"ap`	place le contenu du tampon a après la ligne courante.

5.1.12 Décalage

`>> (ou <<)`	décalage vers la droite ou vers la gauche de toute la ligne courante de m positions, m étant défini par la commande *`:set sw=m`* (par défaut m=8),
`n>> (ou n<<)`	décalage des n lignes suivantes,
`:n1,n2 >> (ou <<)`	décalage des lignes n1 à n2.

5.2 LE MODE COMMANDE DE L'ÉDITEUR EX SOUS VI

L'éditeur ligne *`ex`* est capable de réaliser sur un texte les opérations suivantes :

- recherche d'une chaîne de caractère (motif),
- déplacement et duplication de lignes,
- substitution de chaînes de caractères,
- manipulation de fichier.

L'éditeur *`vi`* passe en mode *`ex`* lorsque l'utilisateur frappe le caractère *`:`* en mode commande *`vi`*.

Le passage en mode *`ex`* est matérialisé par l'apparition du caractère *`:`* sur la dernière ligne de l'écran, indiquant l'attente d'une commande *`ex`*. Cette commande est validée par la touche *`<return>`*.

5.2.1 Listage sélectif et recherche de motif

`:g/motif[ /p]`	liste les lignes contenant *`motif`* dans tout le fichier,
`:m,ng/motif[ /p]`	liste les lignes contenant *`motif`* entre les lignes *`m`* et *`n`*.

5.2.2 Déplacement et duplication de lignes

`:i,jmok`	déplace les lignes *i* à *j* en les mettant après la ligne *k*,
`:i,jcok`	copie les lignes *i* à *j* en les mettant après la ligne *k*.

5.2.3 Substitution de chaînes de caractères

`:s/old/new[ /p]`	substitution du premier *old* de la ligne courante par *new*,
`:s/old/(&)[ /p]`	substitution du premier *old* de la ligne courante par *(old)*,
`:s/old/new/gp`	substitution de tous les *old* de la ligne courante par *new*,
`:i,js/old/new/p`	substitution des premiers *old* des lignes *i* à *j* par *new*,
`:i,jg/old/s//new/p`	idem, mais toutes les lignes modifiées sont listées,
`:i,js/old/new/g`	substitution de tous les *old* des lignes *i* à *j* par *new*,
`:g/old/s//new/g`	substitution des *old* de tout le fichier par *new*,
`:g/old/s//new/gp`	idem, et liste toutes les lignes modifiées,
`:[ n] s[ g]`	répète la plus récente substitution sur la ligne courante ou la ligne numéro *n*,
`:u`	*undo*, annule la plus récente substitution réalisée.

Dans les exemples ci-dessus, i et j peuvent prendre les valeurs suivantes :

- `.` la ligne courante
- *$* la dernière ligne
- *%* tout le fichier

Les expressions régulières utilisées en substitution comportent
/ \ & . * [ ] ^ et **$** (voir chapitre 14).

/	séparateur de chaînes (peut être un caractère quelconque, sauf si risque de confusion),
\	caractère de neutralisation (\& désigne le caractère &),
&	chaîne de caractères à remplacer (la dernière chaîne connue),
•	un caractère quelconque (équivalent à ? dans les commandes de l'interpréteur shell),
*	un nombre quelconque de fois le caractère qui le précède,
[ ]	n'importe quel caractère dans l'intervalle [ ] (ex : [1–4]=[1234]),

`^`	début d'une ligne (ou négation si ce caractère se trouve à l'intérieur et au début de [^…]),
`$`	fin d'une ligne (s/$/./ ajoute un point à la fin de la ligne courante).

5.2.4 Insertion et écriture de fichier

`:n rnom de fichier`	insère le fichier nommé après la ligne *n*,
`:r nom de fichier`	insère le fichier nommé après la ligne courante,
`:r!cmd`	idem, mais avec la sortie de la commande *cmd*,
`:w!nom de fichier`	crée un nouveau fichier avec le contenu du fichier courant,
`:i,jw!nom de fichier`	idem, avec les lignes *i* à *j* du fichier courant,
`:w>>nom de fichier`	ajoute le ficher courant à la fin du fichier nommé,
`:w!nom de fichier`	copie le fichier courant sur le fichier nommé.

5.3 PERSONNALISER VI

Le fichier d'environnement *.exrc* permet de personnaliser le fonctionnement de *vi*. Ce fichier se trouve dans le répertoire d'accueil (HOME). Il contient les commandes permettant de positionner les options de *vi*, et de créer de nouvelles commandes.

5.3.1 Les commandes set

La commande *set* permet de positionner les options de *vi*. Sa syntaxe générale est :

```
:set option[ valeur]
```

Dans *vi*, la commande *:set all* affiche les paramètres positionnés à l'aide de la commande *set*.

Exemples

`:set nu`	permet la numérotation des lignes,
`:set nonu`	enlève la numérotation des lignes,
`:set ai`	permet l'indentation automatique
`:set noai`	enlève l'indentation automatique,
`:set sw=3`	décalage de ligne de 3 caractères,
`:set nomesg`	pas d'intrusion de message dans *vi*,
`:set wrapscan`	lors de la recherche d'une chaîne de caractères, recherche circulaire,
`:set showmode`	indique dans quel mode on se trouve (*Input Mode* ou *Replace Mode*),

`:set showmatch`	positionne pendant une seconde le curseur à la parenthèse correspondant à celle saisie.

5.3.2 Les commandes map

La commande *map* permet de créer de nouvelles commandes sous *vi* par la combinaison de commandes existantes. Sa syntaxe générale est :

```
:map nouvelle_commande ensemble_commandes_vi
```

Cette commande permet d'affecter à une touche spéciale du clavier (qui génère une séquence particulière de caractères) une série de commandes de *vi*. C'est ce que l'on appelle le **mapping**.

Si la séquence de caractères *nouvelle commande* ne comporte qu'un seul caractère, il ne doit pas être numérique. Si cette séquence comporte plus qu'un caractère, elle ne doit débuter ni par un caractère alphanumérique, ni par le caractère *:* (deux points).

Exemples

1) Si la touche du clavier **home** génère le caractère *g*, il est possible d'affecter à cette touche le positionnement en début de fichier à l'aide de la commande *map* ci-dessous.

```
:map g 1G § Le caractère g nous positionne
          § en début de fichier.
```

2) Il est parfois intéressant de rajouter une chaîne de caractères à la suite d'un mot sans quitter le mode commande.

```
:map  @  i@machine.u strasbg.fr<ctrl V><Esc>l
# en saisissant @ en mode commande, vi insère la chaîne
# @machine.u strasbg.fr et repasse en mode commande
```

L'exécution de la commande *:map* sous *vi* permet d'obtenir les nouvelles commandes créées par *map*.

5.3.3 Les commandes map! en mode insertion

La commande *map!* permet de créer de nouvelles commandes sous *vi* par la combinaison de commandes existantes en mode insertion. La syntaxe générale est :

```
:map!  nouvelle_commande  ensemble_commandes_vi
```

Cette commande permet d'affecter à une touche spéciale du clavier (qui génère une séquence particulière de caractères) une série de commandes de *vi* (le mapping) en mode insertion.

Exemple

Lors d'une saisie de texte en mode insertion, on souhaite supprimer la fin de la ligne à l'aide de la touche F2 et poursuivre la saisie. Il est possible d'affecter à cette

touche le basculement en mode commande à l'aide de la touche <Esc>, puis supprimer la fin de ligne (d$), et enfin revenir en mode insertion. La commande *map!* ci-dessous le permet :

```
:map!  <F2>  <ctrl v><Esc>ld$a
```

L'exécution de la commande *:map!* sous *vi* permet d'obtenir les nouvelles commandes créées par *map!*.

On obtient l'insertion d'un caractère spécial (en mode insertion) en le faisant précéder de la combinaison *<ctrl v>*. Par exemple, l'insertion du caractère *<Esc>* est obtenue par *<ctrl v><Esc>*.