Efficacité dans son environnement de travail

> Tools amplify your talent. The better your tools, and the better you know how to use them, the more productive you can be.
>
> Andrew Hunt, The Pragmatic Programmer : From Journeyman to Master

Ce TP tente de donner quelques pistes pour vous permettre de gagner en efficacité dans votre travail au quotidien. Il n'aborde que des outils génériques (gestionnaire de fenêtres, navigateur, terminal, Shell), qui vous seront utiles quelle que soit votre activité. Bien sûr, il est nécessaire d'être efficace avec d'autres outils, comme par exemple SSH.

Le principal conseil à retenir de ce TP est : « réfléchissez à vos habitudes, essayez d'identifier les situations où vous êtes peu efficaces, et tentez d'y remédier ».

1 Clavier et souris

Souris Utiliser la souris est très inefficace. Essayez de vous entrainer à l'utiliser le moins possible. Apprenez les raccourcis clavier de votre gestionnaire de fenêtres. Pour vous entrainer, vous pouvez essayer de désactiver la souris (voir `https://askubuntu.com/questions/199271/x11-disable-mouse`) et de voir combien de temps vous tenez.

Il existe des extensions permettant d'utiliser uniquement le clavier dans votre navigateur Web (Vimium pour Chrome, Vimium-FF pour Firefox).

Clavier Il est important de pouvoir utiliser le clavier de manière relativement fluide. En pratique, une frappe de *secrétaire* avec les huit doigts sur QSDF/JKLM n'est pas forcément la solution la plus efficace, car nous avons souvent besoin de taper des caractères spéciaux. Beaucoup d'informaticiens rapides au clavier n'utilisent pas tous leurs doigts. Par contre, un point indispensable est de savoir taper sans regarder le clavier (car cela vous permet de garder les yeux sur l'écran, et donc de découvrir plus vite vos fautes de frappe). Vérifiez en utilisant votre clavier dans le noir que c'est effectivement le cas ! De nombreux logiciels, sites web ou jeux permettent de s'entrainer à taper plus vite au clavier.

Les cartes de clavier différentes (DVORAK, QWERTY) ne sont pas forcément une solution, car elles sont spécifiques à une langue (français, anglais). En pratique, vous aurez souvent à travailler en plusieurs langues (français, anglais) ou en aucune langue en particulier (Shell, fichiers de configuration).

2 Gestionnaire de fenêtres et environnement graphique

Il existe de nombreux gestionnaires de fenêtres. La mode il y a quelques années était aux *tiling window managers* comme awesome ou xmonad. Ces bonnes idées ont été intégrées en partie dans les gestionnaires de fenêtres classiques (par exemple, voir cette vidéo pour Cinnamon).

Beaucoup de gestionnaires de fenêtres permettent aussi d'utiliser des bureaux virtuels, qui vous permettent de dédier un bureau à une activité.

Pour les terminaux, Terminator permet de faire du *tiling* (mais pour les terminaux uniquement), et permet ensuite de changer de terminal avec des raccourcis clavier (Table 1). Il est aussi possible de définir des profils, puis de lancer Terminator directement avec le profil défini.

Ctrl+Shift+O	Coupe le terminal hOrizontalement
Ctrl+Shift+E	Coupe le terminal VErticalement
Ctrl+Shift+O,P,E,N,E	Coupe le terminal en quatre
Alt+flèches	Se déplace dans un autre terminal

TABLE 1 – Raccourcis clavier de Terminator

3 Shell (bash)

Plusieurs aspects contribuent à être efficace dans le Shell :

— Savoir se déplacer efficacement dans la hiérarchie de répertoires :
  — Penser à utiliser au mieux les chemins absolus ou relatifs (selon les cas)
  — ~ désigne $HOME
  — `cd` (sans paramètre) permet de revenir dans $HOME
  — `cd -` permet de revenir dans le répertoire précédent
— Utiliser au mieux la complétion automatique. Le paquet `bash-completion` fournit de la complétion contextuelle. Par exemple, si on tape `ssh` Tab, alors `bash-completion` permettra de compléter sur des noms de machines, et pas sur des fichiers.
— Utiliser au mieux les raccourcis clavier pour se déplacer et éditer la ligne de commande (voir Figure 1).
— Utiliser au mieux l'historique.

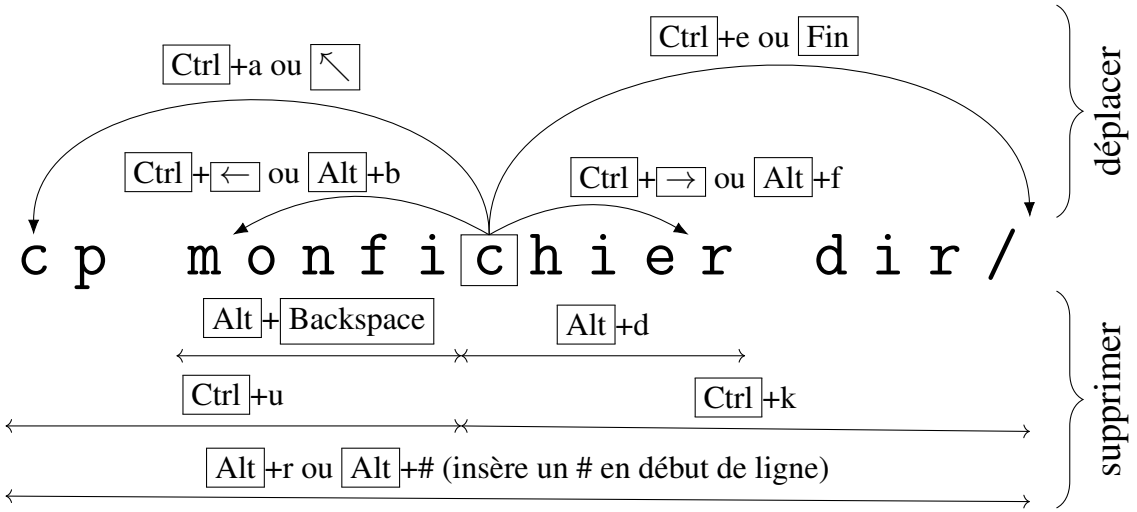

FIGURE 1 – Raccourcis clavier dans Bash, et dans les autres applications utilisant *readline* (inspiré de ce blog). Echap peut être utilisé à la place de Alt. Ctrl +x suivi de Ctrl +e permet de continuer l'édition dans $EDITOR

4 Éditeur (VIM)

VI(M) est probablement l'éditeur le plus populaire chez les administrateurs système. Il a plusieurs avantages :

— Il est installé par défaut sur quasiment tous les systèmes Unix/Linux (parfois dans sa version historique `vi`)
— Il est utilisable à la fois en local (avec une version graphique) et à distance (à travers une connexion SSH, dans un terminal)
— Il propose un large choix de fonctionnalités avancées (coloration syntaxique, nombreux plugins)
— Lorsqu'il est maîtrisé, il permet d'être très efficace

Pour apprendre VIM, le plus simple est d'utiliser `vimtutor`, qui passe en revue toutes les commandes de base.

Quelques rappels :

— Vous pouvez définir une `statusline` dans votre `.vimrc` pour afficher en permanence, par exemple, le numéro de ligne courant. Des plugins comme airline permettent également de redéfinir votre ligne de statut.
— `17gg` permet de se déplacer à la ligne 17.
— `"+p` permet de « coller » du texte qui a été « copié » dans une autre application. De même, `"+y` permet de « copier » du texte pour l'utiliser dans une autre application. (Attention, il faut avoir installé le paquet `vim-gtk3` ou `vim-gnome`. Le paquet `vim` ne permet pas d'utiliser le *clipboard* X).
— La Figure 2 rappelle quelques raccourcis clavier.

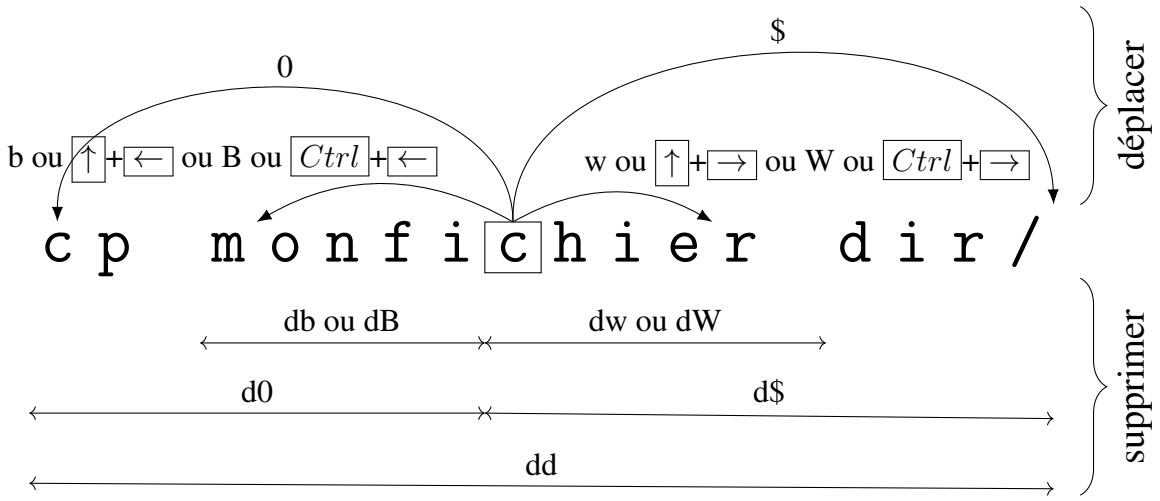

FIGURE 2 – Quelques raccourcis clavier dans `vi`

Une fois les commandes de base acquises avec `vimtutor`, vous pouvez vous perfectionner en jouant à https://www.vimgolf.com/ (et en regardant les commandes utilisées par ceux utilisant moins de coups que vous).

Exemple d'utilisation efficace de VIM

Cet exemple est tiré du challenge One number per line de VimGolf. L'objectif est de passer de :

```
- One number per line -
-----------------------
2,3,5,7,
11,13,17,
19,23,29,
```

à :

```
2
3
5
7
11
13
17
19
23
29
```

Il faut commencer par supprimer les deux premières lignes. C'est faisable avec :

— `2dd`
— `dd.` (`.` répète la dernière modification)
— `2D`
— `dj` (supprime en se déplaçant vers le bas)
— `d`↓ (équivalent à la technique précédente : `j` et ↓ sont équivalents)

Ensuite, il faut *découper* les lignes sur les virgules. Une première solution est de remplacer les virgules dans tout le document par des retours à la ligne, avec `:%s/,/\r/g` . Toutefois, ce n'est pas très pratique, car on se retrouve avec l'état suivant, qu'il faut donc nettoyer pour supprimer les lignes 5, 9 et 13.

```
2
3
5
7

11
13
17

19
23
29

```

On peut s'en sortir un peu mieux en en utilisant `gJ` pour transformer

```
2,3,5,7,
11,13,17,
19,23,29,
```

en :

```
2,3,5,7,11,13,17,19,23,29,
```

À noter qu'il existe aussi `J`, mais qui remplace le retour à la ligne par un espace, ce qui ne nous arrange pas dans ce cas.

Comme tout est sur la même ligne, on peut maintenant utiliser `:s` au lieu de `:%s`. Et il est préférable de supprimer la virgule en fin de ligne avant de faire le remplacement, pour éviter d'introduire une ligne en plus. Une première solution complète en 19 coups est donc :

- `dj` pour supprimer les deux premières lignes
- `3gJ` (ou `gJ.`) pour rassembler les trois lignes
- `$x` pour se déplacer en fin de ligne, puis y supprimer la dernière virgule
- `:s/,/\r/g` pour remplacer les virgules par des retours à la ligne
- `ZZ` pour sortir (ce qui fait gagner un caractère par rapport à `:wq`

Pour faire moins de coups, on peut utiliser une autre fonctionnalité intéressante de Vim : lui faire apprendre, puis rejouer, une séquence d'opérations. Cela se fait avec la commande `q` (voir `:help complex-repeat`). Il faut :

- Démarrer l'enregistrement, en faisant `q` suivi d'un nom de *registre* (`0-9a-z".=*+`). Par exemple `q0`
- Taper une suite de commandes
- Terminer l'enregistrement avec `q`

Pour jouer l'enregistrement, il faudra utiliser `@` suivi du nom de registre. Par exemple, `q0xxq` suivi de `2@0` supprimera 2 fois 2 caractères. (6 caractères seront supprimés au total, avec les 2 supprimés lors de l'enregistrement).

En s'appuyant sur cette technique, on peut donc remplacer les virgules par des retours à la ligne, 10 fois. Il suffit de se déplacer de virgule en virgule (avec `w`, qui permet de se déplacer de *mot* en *mot*), et d'utiliser `r` pour remplacer le caractère.

Une autre solution en 19 coups est donc :

- `dj`
- `3gJ`
- `0` pour aller en début de ligne
- `q0wr<CR>q` pour enregistrer le remplacement d'une virgule par un retour à la ligne (`<CR>` est l'appui sur Entrée)
- `9@0` pour jouer l'enregistrement 9 fois
- `dd` pour supprimer la dernière ligne
- `ZZ`

Pour faire un coup de moins, on peut remarquer que le remplacement de `,` par un retour à la ligne fonctionne même s'il y a un espace après la virgule, grâce à l'indentation automatique. On peut donc se contenter d'utiliser `J` à la place de `gJ` :

- `dj`

— `3J`
— `0`
— `qqwr<CR>q` (en général, on utilise le registre `q` car plus rapide à taper)
— `9@q`
— `dd`
— `ZZ`

5 Astuces diverses

— Le paquet `moreutils` contient des outils supplémentaires (`sponge`, `vidir`, `ts`, ...). Voir https://joeyh.name/code/moreutils/

6 Questions

Q1. Identifiez cinq opérations simples que vous faites habituellement à la souris, et trouvez le raccourci clavier correspondant.

Q2. Si vous utilisez Cinnamon, essayez les différents raccourcis de https://www.lifewire.com/complete-list-of-linux-mint-4064592 (note : Super désigne habituellement la touche avec le logo Windows). Si vous n'utilisez pas Cinnamon, explorez les possibilités de *Tiling* dans votre environnement de bureau.

Q3. Essayez les raccourcis clavier du shell mentionnés dans ce document

Q4. Suivez l'ensemble du tutoriel `vimtutor` pour apprendre VIM

Q5. Sur la base de tout ce que vous avez vu aujourd'hui, faites une liste ordonnée de 10 à 20 actions (couvrant l'ensemble de ce document) dont vous souhaitez vous souvenir pour travailler plus efficacement