R1.04 - TP 1 bonus
vi, vim

Introduction

Nous avons déjà évoqué plusieurs fois quelques raisons d'une bonne maîtrise des outils en ligne de commande. Un petit rappel ?

- Puissance de la ligne de commande par rapport aux outils graphiques : rapidité, flexibilité
- Disponibilité quel que soit l'environnement : local ou distant, interface graphique ou interface texte.

À ce stade de votre apprentissage, vous n'avez pas suffisamment d'expérience pour vous convaincre de ces arguments, et il y en a d'autres que vous découvrirez au fil du temps. Mais vous pouvez faire confiance à ceux qui vous l'enseignent.

S'il ne faut retenir qu'une seule raison de maîtriser la ligne de commande c'est que vous pouvez (lire : "vous allez certainement") avoir besoin d'intervenir, de travailler, de dépanner, d'installer, de configurer des applications, des services sur des machines qui ne sont pas locales (pas devant votre nez) et/ou qui n'ont pas d'interface graphique.

C'est notamment la nature même de ce qu'on appelle un "serveur". Dans votre vie professionnelle (et même à l'IUT), vous serez régulièrement amenés à travailler sur des serveurs.

C'est aussi une contrainte qu'on peut trouver sur des systèmes embarqués dans des objets connectés (IoT : Internet of Things). Régulièrement des étudiants travaillent en stage ou en alternance sur des objets connectés. Par nature, ces objets sont démunis d'interface graphique, on s'y connecte par le réseau pour les configurer, installer et mettre à jour du logiciel que vous pouvez aussi être amené à concevoir et à développer.

Bref, les exemples sont légion.

Éditeur de texte

En TP1, par soucis de simplicité et de rapidité, nous vous avons proposé d'utiliser VSC pour créer des fichiers texte et y saisir du contenu, car les actions qu'on souhaite vous faire expérimenter ne sont pas de créer ou modifier du contenu, mais simplement de

voir le contenu et de manipuler ces fichiers en tant que conteneurs, en tout cas pour le moment.

Cependant, vous serez forcément aussi amenés à créer ou modifier le contenu de fichiers sans disposer d'une interface graphique.

Pour ce faire, vous aurez donc besoin d'un éditeur de texte utilisable dans le Terminal.

Mais un éditeur de texte est plus qu'une simple commande telle que **`cp`** ou **`rm`**. Une différence principale est qu'un éditeur de texte est interactif, il vous permet de saisir du texte, de vous déplacer dans l'espace de saisie, et de faire des actions plus ou moins complexes comme par exemple de rechercher/remplacer des portions dans le texte que vous éditez.

Contrairement à une commande, un éditeur de texte reste en exécution tant que vous ne décidez pas d'en sortir. C'est normal puisqu'on est entré en interaction avec lui, à l'inverse des commandes qui font une tâche (généralement simple) et qui s'arrêtent aussitôt la tâche accomplie.

Un éditeur de texte, même en mode Terminal, s'apparente plutôt à une application, comme pourraient l'être Word, Excel ou même VSC.

Même s'il fonctionne dans le Terminal, un éditeur de texte dispose d'une interface utilisateur, nécessaire pour l'interaction entre lui et l'utilisateur. Attention, qui dit interface utilisateur ne dit pas interface graphique (fenêtre, souris).

Un éditeur dans le Terminal possède une interface semi-graphique, c'est à dire qu'elle utilise les capacités du Terminal à afficher du texte avec des attributs graphiques primitifs : couleur du texte, couleur de l'arrière-plan du texte, ainsi que certains caractères graphiques (barres verticales, barres horizontales, angles, croisement etc.) permettant de faire quelque formes très simples (rectangles, grilles, petits menus etc.)

Nous allons voir 2 éditeurs en mode Terminal (enfin, surtout un).

vi et vim

Le premier éditeur est **`vi`** (prononcer "vi-aille"), qui signifie(rait ?) "Visual Instrument". C'est l'éditeur de texte historique d'Unix. C'est aussi le cauchemar de l'étudiant.e et c'est pour cela qu'on vous a poussé du côté de VSC... pour le moment... 😈

C'est un éditeur que vous êtes quasiment certain de pouvoir trouver sur n'importe quel Linux, à moins qu'il ait subi une grosse cure de minceur, ce qui peut arriver sur des environnements contraints en espace de stockage. De toute façon, si vous n'avez pas **`vi`**, vous n'aurez certainement rien d'autre comme éditeur de texte !

Cet ancêtre qu'est **`vi`** possède une version améliorée qui s'appelle **`vim`** (**`vi`** *improved*).

Par simplicité, on nomme souvent ces deux versions par le même nom, juste : **`vi`**

Bonne nouvelle, on va utiliser cette version améliorée !

Mauvaise nouvelle, elle n'est pas plus "user friendly" que l'originale ! Elle est *improved* car elle a des fonctionnalités supplémentaires. Bon, elle est quand même un tout petit peu plus agréable à utiliser, mais ne vous emballez pas trop quand même.

Premier essai

Attention à bien suivre les instructions sous peine de rester bloqué…

Ne tapez que ce qui est demandé, au caractère près, sans chercher à saisir autre chose, même si le résultat est inattendu ou que vous avez l'impression que ça ne fonctionne pas comme vous le pensiez.

Ouvrez un Terminal et placez-vous dans un dossier de travail (**`cd`**, **`mkdir`** etc.)

Maintenant lancez cette commande :

```
vim fic.txt
```

Vous devez vous retrouver avec votre fenêtre intégralement vidée de son contenu, une colonne de tilde (**~**) sur la gauche du Terminal et une ligne en bas qui affiche ceci : **`"fic.txt" [New File]`**

Observez maintenant l'affichage, pendant que vous tapez ceci doucement, caractère par caractère :

```
toto
```

Est-ce que vous avez obtenu un affichage ? Avez-vous le mot : **`toto`** qui apparaît dans votre espace de texte ?

En principe **`vi`** est resté indifférent à ce que vous avez tapé, c'est normal (si si, on va l'expliquer un peu plus loin) et rien ne s'est affiché. Si ce n'est pas le cas, faites appel à votre enseignant.e.

Refaisons un autre test. Toujours en observant l'affichage, tapez maintenant ceci doucement, caractère par caractère :

```
lili
```

Alors, qu'avez-vous obtenu cette fois-ci ?

En principe, vous devriez avoir uniquement une seule fois les 2 lettres : **li**

C'est un comportement qui doit vous sembler très étrange et qui va l'être encore plus si maintenant vous essayez de taper encore une fois :

> **toto**

Est-ce que, cette fois-ci, vous avez un affichage du texte **toto** ?

En principe vous avez obtenu le texte **toto** à la suite du texte **li** précédent.

Les 3 modes

Expliquons ce qui vient de se passer.

Contrairement aux éditeurs de textes auxquels vous êtes certainement habitué, dans **vi** on ne peut pas saisir du texte directement sans le prévenir au préalable qu'on souhaite le faire, car **vi** dispose de 3 modes de fonctionnement, que nous introduisons ci-dessous et que nous détaillerons mieux un peu plus bas :

- Un mode "insertion" : c'est le mode habituel dans un éditeur de texte mais ce n'est pas le mode par défaut au lancement de **vi**. Il permet de saisir du texte.
- Un mode "commande" : c'est le mode par défaut au lancement de **vi**. On l'appelle aussi le mode "normal", même si on s'accordera pour dire qu'il n'y a que **vi** pour trouver ce fonctionnement "normal" !

  C'est un mode qui permet de faire des actions sur le texte, telles que copier/couper/coller du texte, ou de faire des actions sur l'éditeur **vi** lui-même, telles que le basculer en mode "insertion".
- Un mode "ligne de commande" : à ne pas confondre avec le mode "commande", il permet aussi de faire des actions sur le texte et est complémentaire du mode "commande". Pour différencier ces 2 modes ("commande" et "ligne de commande") on peut dire que le premier permet de saisir des commandes plus rapidement que le second. On va voir quelques exemples un peu plus loin.

Explications du premier essai

Revenons maintenant à notre premier essai pour expliquer ce qui s'est passé.

Pendant ces explications, ne tapez rien dans **vi** avant d'y avoir été invité.

On vient de voir que le mode par défaut, au lancement de **vi**, est le mode "commande" et non pas le mode "insertion" qui permettrait de saisir du texte.

On vient d'introduire le mode "commande" comme étant un mode permettant de faire des actions sur le texte ou sur l'éditeur **vi** lui-même.

Les actions du mode “commande” se font par simple appui sur une des touches du clavier : 1 touche = 1 action immédiate, sans avoir à faire quoi que ce soit d’autre, ni avoir à appuyer sur la touche **ENTRÉE**.

Par exemple, pour passer **vi** en mode “insertion”, ce qui nous permettra alors de saisir du texte, on peut utiliser au choix :

- La touche **i** (pour “**i**nsert”) :
- La touche **a** (pour “**a**ppend”)
- La touche **R** (pour “**R**eplace”)

et plusieurs autres encore. Notez déjà qu’il y a une différence entre les minuscules et les majuscules en mode “commande”.

Maintenant qu’on vient de découvrir quelques “commandes” pour passer en mode “insertion”, on va enfin comprendre ce qui s’est passé quand on a saisi les lettres du mot **toto** (sans effet apparent) puis du mot **lili**. (avec pour effet l’affichage de **li** seulement) et enfin d’un second **toto** (affiché complètement cette fois-ci).

Alors qu’il était encore en mode “commande”, **vi** a interprété chaque lettre de **toto** comme autant de commandes et non pas un texte à ajouter au texte déjà présent. A priori, aucune des lettres tapées (**t** et **o**) n’ont eu d’effet, en tout cas aucune n’a permis de passer **vi** en mode “insertion”.

Puis **vi** a fait de même avec chaque lettre de **lili**. et cette fois-ci, à la saisie de la première occurrence de la lettre **i**, **vi** a identifié ce **i** comme une commande (“**i**nsert”) lui ordonnant de passer en mode “insertion”, ce qui a eu pour effet que tout ce qui a été tapé ensuite a été considéré, non plus comme des commandes, mais comme étant du texte à ajouter à la suite du texte existant, d’où l’affichage de **li** ainsi que du second **toto** juste après.

A partir de là, **vi** étant passé en mode “insertion”, tout ce que vous allez saisir sera du texte et apparaîtra à l’écran au fur et à mesure de votre saisie.

Maintenant, vous pouvez essayer en tapant 2 ou 3 lignes de texte, avec des retours à la ligne (touche **ENTRÉE**).

Se déplacer en mode “insertion”

Dans ce mode “insertion”, pour vous déplacer dans votre texte, vous pouvez utiliser les flèches du clavier (⬅ ⬆ ⬇ ➡). Testez.

Sortir du mode “insertion”

Maintenant que vous êtes passé en mode “insertion”, vous vous retrouvez certainement plus à l’aise car c’est le comportement classique de la plupart des autres éditeurs que vous connaissez (Geany, VSC, etc.).

Cependant, ce mode “insertion” est limité exclusivement à la saisie du texte. Dès que vous aurez besoin de faire une action telle que sauver votre fichier, copier/coller des lignes, rechercher quelque chose dans le texte etc., vous devrez quitter ce mode “insertion”, pour repasser en mode “commande”.

Pour ce faire, vous devez utiliser la touche **`ESC`** (ou **`ECH`** sur certains claviers), qui est généralement située dans le coin supérieur gauche du clavier. Cette touche signifie/s’appelle **`ESCAPE`** ou **`ECHAPPE(MENT)`** et porte bien son nom dans le cas présent : elle permet d’échapper au mode “insertion”, et revenir ainsi au mode “commande”.

Essayez.

Pour vérifier que ça fonctionne, tentez ensuite de saisir à nouveau le texte :

> **`toto`**

On est revenu à la situation initiale, en principe : rien ne s’affiche, on est en mode “commande”.

Vous pouvez refaire quelques essais :

- Repassez en mode “insertion” par l’appui sur la touche **`i`**
- Saisissez du texte
- Ressortez avec **`ESC`**

Détails sur l’entrée en mode “insertion”

On a vu précédemment que les touches **`i`**, **`a`** et **`R`** permettent de passer en mode “insertion”. Voici précisément ce qui les différencie :

- **`i`** : insertion du texte là où se trouve le curseur (**`i`**nsert = insertion)
- **`a`** : insertion du texte à partir de la position juste après le curseur (**`a`**ppend = ajout)
- **`R`** (**`r`** majuscule) : remplacement du texte déjà présent à partir de l’endroit où se trouve le curseur. La saisie peut être plus longue que le texte actuel. (**`R`**eplace = remplacement)

Pour être plus précis, au lieu de parler des touches, on devrait plutôt dire les lettres **`i`**, **`a`** et **`R`** en respectant bien s’il s’agit d’une minuscule ou d’une majuscule, c’est très important. Ainsi, on peut ajouter ces autres possibilités :

- **`I`** (**`i`** majuscule) : insertion du texte au début de la ligne sur laquelle se trouve le curseur
- **`A`** (**`a`** majuscule) : insertion du texte à la fin de la ligne où se trouve le curseur
- **`r`** (minuscule cette fois-ci) : remplacement du caractère sous le curseur par le prochain caractère qu’on va saisir. Attention, vous n’entrez pas en mode

"insertion" avec cette action-là : après la saisie du caractère de remplacement, vous restez en mode "commande".

Choisissez ce qui vous convient le mieux en fonction de la situation.

Amusez-vous à tester toutes ces possibilités pour vous familiariser.

Se déplacer en mode "commande"

Dans le mode "commande", pour vous déplacer dans votre texte, vous pouvez aussi utiliser les flèches du clavier (⬅ ⬆ ⬇ ➡). Testez-les.

A noter que si ça ne fonctionne pas et que l'utilisation des flèches provoque l'insertion de caractères bizarres dans votre texte, c'est certainement que vous êtes sous **`vi`** et non pas sous **`vim`**.

Certains systèmes sont configurés pour lancer **`vim`** même avec la commande **`vi`**, mais pas tous.

Certains systèmes n'ont tout simplement pas de **`vim`** mais simplement un **`vi`** de base.

Il est aussi possible de se déplacer sans utiliser les flèches, en mode "commande" :

- **`h`** : remplace la flèche ⬅
- **`k`** : remplace la flèche ⬆
- **`j`** : remplace la flèche ⬇
- **`l`** (**`L`** minuscule) : remplace la flèche ➡

La barre d'espace permet aussi de se déplacer sur le caractère suivant mais ne va plus loin que le bout de la ligne (comme pour le **`l`** d'ailleurs).

Pour passer à la ligne suivante, on peut utiliser la touche **`ENTRÉE`**.

En restant en mode "commande", testez toutes ces possibilités pour vous familiariser.

Autres déplacements en mode "commande"

Jusqu'ici le mode "commande" nous a semblé un peu bizarre, archaïque et compliqué pour pas grand chose.

Attardons-nous maintenant sur des actions de déplacement qui vont donner de l'intérêt à ce mode "commande". Attention, vous restez en mode "commande" après l'action. Ce n'est juste qu'un déplacement du curseur.

- **`w`** : se déplace d'un mot vers l'avant (**`w`** : word)
- **`b`** : se déplace d'un mot en arrière (**`b`** : backward)
- **`$`** : se déplace jusqu'à la fin de la ligne courante
- **`0`** (chiffre zéro) : se déplace au début de la ligne courante
- **`G`** : se déplace à la dernière ligne

- **un nombre** suivi de **G** : se déplace à la ligne dont le numéro est indiqué par le **nombre**, ou à la dernière ligne s'il y a moins de lignes. Exemple : **123G**

  Pour se rendre sur la 1ère ligne : **1G**

Le mode "ligne de commande"

vi possède un autre mode pour effectuer des actions, le mode "ligne de commande".

Alors que le mode "commande" est un mode d'actions rapides et basiques : la plupart du temps il ne suffit que d'une seule touche pour faire l'action (**a**, **i**, **$**, **0**, **w**, **b**), le mode "ligne de commande" permet des actions plus sophistiquées qui nécessitent donc plus de paramètres pour décrire de ce qu'on veut faire.

De ce fait, l'usage d'une simple touche n'est pas envisageable et **vi** a donc introduit un mode "ligne de commande" qui, comme la ligne de commande dans un Bash, permet de décrire l'action en suivant une syntaxe particulière.

Pour entrer en mode "ligne de commande", il faut taper le caractère **:** (deux-points), ce qui fait apparaître une ligne en bas de l'écran, la fameuse "ligne de commande" de **vi** !

Attention, cette ligne de commande n'a rien à voir avec celle du Bash. Sa syntaxe est propre à **vi**.

Commençons par un petit exemple. Rappel : vous devez d'abord être en mode "commande" (appuyez sur **ESC** si ce n'est pas le cas). Puis tapez ceci en appuyant sur **ENTRÉE** pour exécuter la commande :

```
:set number
```

Ainsi, le **:** fait passer en mode "ligne de commande" pour pouvoir saisir et exécuter le **set number**.

Cette commande provoque l'affichage des numéros de ligne dans l'éditeur.

Vous pouvez faire l'inverse pour les masquer :

```
:set nonumber
```

A chaque exécution d'une commande, vous revenez en mode "commande". On entre en mode "ligne de commande" uniquement quand on tape un **:**

Pour sortir du mode "ligne de commande" sans exécuter de commande, il suffit de supprimer la ligne (y compris le caractère **:**), ou encore de taper simultanément les touches **CTRL**+**C**.

Quitter vi

Avant d'entreprendre plus de choses, il est temps d'apprendre comment quitter **vi**.

Pour quitter, vous devez obligatoirement être en mode "commande".

Comme souvent avec **vi**, plusieurs manières vous sont offertes pour faire les choses.

En voici quelques unes pour quitter :

- **:x** : quitte en sauvant les modifications apportées au fichier (e**x**it). Vous aurez noté qu'il s'agit d'une commande en "ligne de commande"
- **:q** : quitte (**q**uit) sans sauver mais à condition que le fichier n'ait pas été modifié. Dans le cas contraire, **vi** refusera de quitter.
- **:q!** : quitte sans sauver, même si le fichier a été modifié ! Le **!** (point d'exclamation) est là pour indiquer qu'on sait très bien ce qu'on veut faire. **vi** ne dira rien et ne refusera pas de quitter. Évidemment, les modifications faites dans le fichier seront perdues.
- **ZZ** (attention, c'est en majuscule) : quitte en sauvant. Vous noterez qu'il s'agit d'une commande directe (en mode "commande") et qu'elle nécessite 2 touches et non pas une seule comme la plupart des autres commandes.

Il est possible de combiner des commandes en les juxtaposant.

Ainsi, la commande **:w** permet de sauver (**w**rite) le fichier sans quitter **vi**.

Mais en combinant avec **:q** on peut, en une seule commande, sauver puis quitter, de cette façon :

- **:wq** : sauve le fichier (**w**) puis quitte (**q**)

Pour terminer avec le **:w**, on peut aussi sauver le fichier sous un autre nom, de cette façon :

- **:w toto** : sauve le fichier (**w**) sous le nom **toto**, et reste dans l'éditeur. Attention, le fichier **toto** ne doit pas exister déjà sinon **vi** refusera.
- **:w! toto** : sauve le fichier sous le nom **toto**, en écrasant un éventuel ancien fichier **toto** s'il existe. Le **!** est encore là pour indiquer qu'on sait ce qu'on demande. De façon générale, on retrouvera toujours cette signification avec le **!**

Pour vous familiariser avec ces commandes, faites des tests avec chacune d'elles.

Comme vous quittez l'éditeur à chaque fois, vous devez évidemment relancer une comme **vi** pour le test suivant.

Recherche de texte

Éditer du texte, notamment quand on code, nécessite souvent de rechercher des occurrences. Pour les tests suivants, nous allons lancer **vi** sur un gros fichier.

Avant tout, il va vous falloir créer ce gros fichier. On aurait pu le faire directement dans **vi**, mais pour gagner du temps, vous allez le créer avec VSC.

Pour l'opération suivante, il est conseillé de charger, même temporairement, votre PDF dans un onglet de votre navigateur Web car la visionneuse PDF sous Linux (à l'IUT) ne permet pas (???) de sélectionner du texte.

Commencez par sélectionner et copier (**CTRL**+**C** ou par le menu) le contenu de la 1ère page de ce PDF, puis dans VSC, créez un fichier **bigfic**, collez-y ce que vous venez de copier et sauvez votre fichier.

Vous pouvez maintenant quitter VSC.

Dans un Terminal, déplacez vous (**cd**) là où vous avez créé **bigfic**, puis tapez ceci :

```
vim bigfic
```

Vous devez donc y voir le texte collé. Si ce n'est pas le cas ou que vous n'y arrivez pas, faites appel à votre enseignant.e.

Tout ce qui suit fonctionne uniquement en mode "commande" !

Pour rechercher une occurrence d'un texte, vous allez entrer dans un mode spécial en appuyant sur la touche **/** (slash), ce qui va afficher une ligne en bas, comme quand vous entrez en mode "ligne de commande" avec le caractère **:**

Ce que vous allez taper à la suite du **/** , jusqu'à appuyer sur la touche **ENTRÉE**, sera le texte que vous allez chercher.

Cherchez le mot **que** :

```
/que
```

Le curseur se place sur la 1ère occurrence trouvée, en partant de la position actuelle du curseur.

Pour trouver la suivante, appuyez sur la touche **n** (en minuscule, pour **n**ext)

Pour trouver la précédente, appuyez sur la touche **N** (en majuscule)

Faites plusieurs essais, vous remarquerez que vous cherchez un **que** n'importe où dans le texte, indifféremment en tant que mot ou en tant que partie de mot. C'est donc un abus de langage de dire qu'on cherche le "mot **que**". **vi** cherche la suite des 3 lettres consécutives **q**, **u** et **e**, sans aucune notion réelle de mot.

Pour effectuer une recherche en sens inverse dès le départ, utilisez un **?** à la place du **/** :

> **?vous**

En cherchant l'occurrence "suivante" (**n**) vous aurez donc la précédente dans l'ordre d'apparition dans le texte et pareil pour la précédente (**N**) qui sera ainsi la suivante dans le texte

La recherche boucle en début ou en fin de fichier en fonction du sens de la recherche.

Remplacer du texte

Pour remplacer une occurrence d'un texte par un autre texte, vous allez utiliser le mode "ligne de commande", celui avec le caractère **:** , comme vous le savez.

La syntaxe est composée ainsi :

- La portée de la recherche, c'est à dire sur quelle portion du texte on souhaite opérer le remplacement.
- Le caractère **s** pour indiquer une action de remplacement (pour **s**ubstitute)
- Un caractère quelconque qui va servir de séparateur entre ce qu'on cherche et ce par quoi on remplace. Généralement on utilise un **/** mais ça peut être n'importe quel caractère qui n'apparaît pas dans le texte que vous cherchez ni dans celui qui servira de remplacement.
- Le texte à chercher
- Le même caractère de séparation (donc sans doute un **/**)
- Le texte à remplacer

Enfin, de manière optionnelle, vous pouvez ajouter ceci à la fin :

- Le même caractère de séparation (donc sans doute un **/**)
- Un **g** pour signifier qu'on souhaite remplacer toutes les occurrences de chaque ligne (pour **g**lobal), sinon ce sera uniquement la 1ère occurrence sur chaque ligne
- Un **i** pour signifier qu'on ne différencie pas les minuscules des majuscules (pour **i**gnore case)

Avant de voir un exemple, il faut évoquer la notion de "portée".

Voici quelques portées possibles :

- **%** : le fichier dans son intégralité
- **un nombre** : la ligne portant ce numéro
- **un nombre,un autre nombre** : les lignes numérotées entre ces 2 nombres

Prenons maintenant cet exemple commenté :

> **:1,5s/que/***Q U E***

Sur chacune des 5 premières lignes (**1,5**), on va remplacer (**s**) toutes les 1ères occurrences de **que** (en respectant les minuscules) par le texte *****Q U E*****.

Testez-la et vérifiez que :

1) Ca fonctionne
2) Seules les 5 premières lignes ont été affectées
3) Seules les 1ères occurrences de chacune des lignes ciblées ont été affectées

Profitons-en pour apprendre comment annuler une commande passée : tapez **u** (pour **u**ndo) et vos remplacements disparaissent. Vous pouvez effectuer plusieurs annulations successives pour revenir plusieurs actions en arrière.

Notez que pour refaire, c'est à dire pour annuler l'annulation, vous pouvez utiliser la commande **CTRL**+**R** (pour **r**edo)

Essayons un autre exemple maintenant :

> **:%s/e/EUH!!!/gi**

Sur l'intégralité du fichier (**%**), on va remplacer (**s**) sur chaque ligne toutes les occurrences (**g**) de la lettre **e**, qu'elle soit en minuscule ou en majuscule (**i**), par le texte **EUH!!!**

Testez-la et vérifiez la cohérence du résultat.

Vous pouvez annuler aussi cette action.

À vous de jouer maintenant. Repartez toujours du texte initial en annulant votre action précédente ou en quittant et ouvrant de nouveau le fichier **bigfic**. Attention il peut y avoir des pièges :

- Remplacez toutes les virgules par des espaces
- Remplacez toutes les virgules par des **.** (points) sur les 3 dernières lignes
- Remplacez tous les espaces de la ligne 1 par rien.
- Remplacez tous les **/** par le texte **UN_SLASH**
- Remplacez les **(** par des **[** et les **)** par des **]**

Que se passe-t-il pour la dernière question ?

Copier, couper, coller

Pour terminer, voyons comment on peut copier, couper et coller des portions de texte.

Ces actions sont des commandes, vous devez donc être en mode "commande".

Voici quelques exemples pour copier des lignes entières de texte dans le presse-papier :

- **yy** : copie (pour **y**ank) la ligne courante (celle où se trouve le curseur)

- **nombre** suivi de **yy** : copie **nombre** lignes à partir de la ligne courante. Exemple : **3yy** copie 3 lignes.

Ces exemples copient des portions de ligne :

- **y$** : copie de la position du curseur jusqu'à la fin de la ligne (saut de ligne exclu)
- **Y** : fait pareil
- **yw** : copie de la position du curseur au début du mot suivant
- **yiw** : copie le mot dans lequel se trouve le curseur.

La copie laisse le texte intact.

On peut aussi couper du texte. A l'inverse de la copie, la coupe fait une copie et retire ce qui est copié du texte d'origine. La syntaxe est exactement la même que pour copie, en utilisant la lettre **d** (pour **d**elete) à la place du **y**.

Ainsi, on aura :

- **dd** : coupe la ligne courante
- **nombre** suivi de **dd** : coupe **nombre** lignes à partir de la ligne courante. Exemple : **3dd** coupe 3 lignes.
- **d$** : coupe de la position du curseur jusqu'à la fin de la ligne (saut de ligne exclu)
- **D** : fait pareil
- **dw** : coupe de la position du curseur au début du mot suivant
- **diw** : coupe le mot dans lequel se trouve le curseur.

Comme pour la copie, il existe bien d'autres combinaisons possibles. Consultez la doc (voir les cheat sheets en fin de TP)..

Une fois que du texte a été copié ou coupé, il peut être collé là où se trouve le curseur. Positionnez-vous au bon endroit avant de coller :

- **p** (minuscule) : colle (pour **p**aste) après le curseur
- **P** (majuscule) : colle avant le curseur

Il existe aussi une version "ligne de commande" avec la syntaxe suivante :

- Portée (voir dans le paragraphe **Remplacer du texte**)
- La lettre **y** pour copier ou la lettre **d** pour couper

Exemple :

```
:5,7d
```

pour couper les lignes **5**, **6** et **7**.

```
:%y
```

pour copier tout le texte.

Portée améliorée

On a vu que la portée d'une action, comme un copier ou un couper, peut être définie avec un **%** ou un intervalle de numéros de ligne.

Il y a une autre façon pratique de procéder quand on veut faire une action de la position courante du curseur jusqu'à une occurrence d'un texte.

On sait rechercher l'occurrence d'un texte avec la syntaxe **/le_truc_qu_on_cherche**.

Pour copier le texte jusqu'à cette occurrence, il suffit de taper un seul **y** (pour indiquer qu'on va copier quelque chose) et de faire immédiatement une recherche avec **/quelque_chose**. En même temps que la recherche va se faire, **vi** copiera tout ce qui se trouve entre la position initiale du curseur et sa position finale une fois l'occurrence trouvée. A noter que, dans cette configuration, le curseur restera quand même à sa position initiale et non pas à la position de l'occurrence, même s'il a fait sa recherche et a trouvé l'occurrence.

Testez ceci (sans jamais appuyer sur **ENTRÉE** sauf si indiqué) :

> **1G**
> **y/Bref** (+ appuyer sur ENTRÉE)
> **G**
> **p**

Vous devriez ainsi :

1) Vous déplacer en tout début de fichier
2) Copier les lignes jusqu'au mot **Bref**
3) Vous déplacer sur la dernière ligne
4) Coller le texte copié en fin de fichier

Cheat sheet

Voici quelques cheat sheets pour vous aider à aller plus loin dans votre maîtrise de **vi** :

- https://vim.rtorr.com
- https://devhints.io/vim
- https://vimsheet.com

Sinon il y a toujours le Discord de votre enseignant.e système préféré.e...

Epilogue

Vous êtes maintenant armé pour faire des actions de base sur n'importe quel fichier texte avec **`vi`** !

Vous êtes sur la voie pour entrer dans le cercle des barbus !

Nano

Un autre éditeur de texte en mode Terminal s'appelle **`nano`**. Il possède une petite interface semi-graphique.

Il est beaucoup plus intuitif que **`vi`**, mais même s'il est assez courant, il ne figure pas parmi les outils que vous êtes quasiment sûr de trouver sur tous les systèmes.

Notre étude de cet éditeur va donc se résumer au strict minimum.

Pour éditer un fichier avec **`nano`** :

```
nano bigfic
```

Pour se déplacer dans le texte, on utilise les flèches, et pour saisir du texte, il suffit de taper directement le texte.

Les principales commandes sont affichées en bas de la fenêtre et il faut juste savoir que le caractère **`^`** signifie : la touche **`CTRL`**. Ainsi **`^X`** signifie l'appui simultané sur **`CTRL`**+**`X`**

Vous disposez d'une aide en ligne en tapant **`CTRL`**+**`G`**

Cet éditeur est bien plus accessible aux novices que ne l'est **`vi`**. Aussi nous n'irons pas plus loin dans les explications, vous devriez rapidement vous en sortir par vous-même.

Conclusion

Maîtriser les bases d'un éditeur en mode Terminal est essentiel pour votre avenir professionnel.

Ne sous-estimez pas son intérêt ni la puissance cachée derrière une interface austère.

On ne vous demande pas de passer votre vie à coder avec, vous aurez le droit d'utiliser des éditeurs conviviaux et graphiques quand ce sera possible (hélas pas en R1.04 - Systèmes ! 😈), mais vous ne devez pas découvrir son existence, ni son maniement, le jour où vous aurez à intervenir sur un serveur distant, surtout s'il s'agit d'une urgence vitale !