Les éditeurs de texte *standards*

Éditer (c'est-à-dire essentiellement *modifier*) des fichiers textes est une des activités les plus fréquentes que vous allez avoir à faire lors de votre carrière d'informaticien. Pour cela vous allez avoir à utiliser un type de programme particulier : un éditeur de texte. C'est un programme qui manipule le contenu de fichiers respectant *la* convention texte. Il existe de très nombreux éditeurs de textes, mais trois d'entre eux sont un peu à part et vous allez apprendre à les utiliser.

Le programme `vi` est un éditeur de texte *visuel* se basant sur l'éditeur de texte `ed`. Il est apparu dans les premières versions des implémentations d'Unix faites par l'Université de Californie à Berkeley[1]. Il est loin d'être un éditeur très évolué et encore moins convivial[2]. Néanmoins, comme `ed`, c'est désormais un standard sur toutes les versions d'UNIX, c'est-à-dire que quelque soit la version d'UNIX que vous utiliserez la commande `vi` sera certainement présente. Il est très puissant et ne nécessite que très peu de ressources.

La majorité des directives de `vi` et de `ed` sont utilisables dans d'autres commandes UNIX comme par exemple `more`, `less`, `sed`, etc. Toutes ces commandes sont elles-aussi basées sur l'éditeur en ligne `ed`.

Il existe aujourd'hui un grand nombre d'implémentation de `vi` toutes différentes, cependant elles respectent un fonctionnement minimum commun que nous allons étudier dans ce TP et qui correspond aux premières versions historiques de `vi`. Pour plus de détails sur `vi` vous pourrez vous reporter à la *Vi Lovers Home Page* ou à la page de **`vim`** (une implémentation très répandue).

Emacs est l'éditeur standard du système GNU (une partie importante des distributions GNU/Linux) et, comme `vim`, est disponible sur la plupart des versions d'Unix ou de Windows. Il nécessite plus de ressources que `ed` ou `vi` mais offre un très grand nombre de fonctionnalités, comme la possibilité d'être étendu via des *plugins* écrits en LISP. Pour plus de détails sur Emacs vous pourrez vous reporter à sa page principale.

L'éditeur de texte en ligne `ed`

`ed`[3] est un programme rudimentaire permettant de créer, voir ou éditer des fichiers textes, c'est-à-dire des fichiers contenant des lignes de caractères terminées par un passage à la ligne. `ed` ne sait manipuler que les lignes de tels fichiers. Il fonctionne sur tous les terminaux possédant un canal d'entrée et un canal de sortie sans aucune autres contraintes. Comme on peut le lire avec délice dans Ed Mastery, `ed` est certes rudimentaire mais offre un nombre de fonctionnalités suffisant pour le rendre aussi puissant, sinon plus, que bien nombre des éditeurs tape à l'œil d'aujourd'hui.

`ed` peut être appelé avec ou sans argument. Si un argument est passé à `ed` il considèrera que cet argument est un chemin valide vers un fichier texte. Dans ce cas il lira le contenu du fichier et le copiera en mémoire afin de pouvoir l'utiliser. Si le fichier n'existe pas un message d'erreur (le simple caractère `?`) sera envoyé mais le chemin sera conservé pour les opérations suivantes. Aucun enregistrement des modifications faites sur le contenu en mémoire ne sera effectué sans une demande explicite de l'utilisateur.

`ed` possède 2 modes de fonctionnement :

— le mode **commande**, les caractères arrivant à `ed` par le canal d'entrée du terminal sont considérés comme des ordres de manipulation du contenu en cours d'édition ;
— le mode **saisie**, les caractères arrivant à `ed` par le canal d'entrée du terminal sont considérés comme devant être ajoutés au contenu qui est en mémoire.

Lors de son démarrage `ed` est en mode **commande**. Le passage en mode de **saisie** se fait après réception de certaines commande (`a`, `i`, `c` par exemple). Pour quitter le mode de **saisie** il suffit d'envoyer une ligne ne contenant que le caractère `.`.

Une commande `ed` peut être préfixée par des caractères désignant une ligne, ou un bloc de lignes, sur laquelle la commande doit être exécutée. On appelle cela l'**adresse** sur laquelle la commande doit s'appliquer. Certains caractères ou mots désignent des lignes spécifiques :

— un nombre *n* correspond à la *n*-ième ligne en mémoire. La première ligne en mémoire est la ligne 1.
— `.` désigne la ligne courante (c'est-à-dire la dernière ligne sur laquelle a eu lieu une opération).
— `$` désigne la dernière ligne du contenu en mémoire.
— `/motif/` correspond à la prochaine ligne contenant le mot *motif* (à partir de la ligne courante).
— `?motif?` correspond à la précédente ligne contenant le mot *motif* (à partir de la ligne courante).

1. Ces implémentations ont données naissance à la famille des Unix BSD. Pour un historique complet des implémentations d'UNIX reportez-vous à l'adresse `http://www.levenez.com/unix`
2. ...pour les utilisateurs habitués à utiliser plus souvent une souris que leur cerveau. Mais tout ça n'est qu'une question d'habitude.
3. Initialement écrit par Ken Thompson et Dennis Ritchie, souvent remplacé par `ex` dans les implémentations récentes d'UNIX.

Un bloc de lignes est représenté par la ligne de départ et la ligne de fin séparée par le caractère `,`. Par exemple, le bloc `1,$` représente l'ensemble du contenu en mémoire. Quand `ed` reçoit une commande qu'il ne comprends pas il envoie sur le canal de sortie du terminal le caractère `?`.

Le tableau 1 rassemble un certain nombre des commandes `ed` importantes. La page du manuel et surtout *info* de `ed` vous en dira plus.

Commandes `ed`	
`P`	Affiche un *prompt*.
`h`	Explique la dernière erreur.
`u`	Annuler la dernière commande.
⟨n⟩	Fixer la ligne courante comme étant la ligne ⟨n⟩
`(.,.)p`	Afficher les lignes spécifiées.
`(.,.)n`	Afficher les lignes spécifiées en les numérotant.
`(.)a`	Ajouter le texte saisi après la ligne spécifiée. *fait entrer en mode saisie*
`(.,.)c`	Supprimer les lignes spécifiées et remplace les par le texte saisi. *fait entrer en mode saisie*
`(.,.)d`	Supprimer les lignes spécifiées.
`(.)i`	Ajouter le texte saisi avant la ligne spécifiée. *fait entrer en mode saisie*
`(.,.)m(.)`	Déplacer les lignes spécifiées après la ligne dont la spécification suit le caractère `m`.
`(.,.+1)j`	Joindre les lignes spécifiées en une seule et même ligne.
`(.,.)s/`⟨motif1⟩`/`⟨motif2⟩`/g`	Remplacer toutes les occurences de ⟨motif1⟩ par ⟨motif2⟩ sur les lignes spécifiées.
`(.,.)t(.)`	Copier les lignes spécifiées après la ligne dont la spécification suit le caractère `t`.
`(.,.)y`	Copier dans un tampon les lignes spécifiées.
`(.)x`	Coller le contenu du tampon après la ligne spécifiée.
`e` ⟨chemin⟩	Lire le fichier ⟨chemin⟩ et copier son contenu en mémoire.
`($)r` ⟨chemin⟩	Lire le fichier ⟨chemin⟩ et insérer son contenu après la ligne spécifiée.
`(1,$)w`	Enregistrer les lignes spécifiées dans le fichier courant.
`(1,$)w` ⟨chemin⟩	Enregistrer les lignes spécifiées dans le fichier ⟨chemin⟩.
`!` ⟨commande⟩	Exécute ⟨commande⟩ dans un shell.
`q`	Quitter `ed`

TABLE 1 – **Commandes** `ed`. Les commandes sont montrées ici avec leur adresse d'application par défaut. Ces adresses par défaut sont celles que la commande considère si aucune adresse n'est spécifiée. Ces valeurs par défaut sont placées entre parenthèses uniquement pour simplifier leur identification. **Les parenthèses ne doivent pas être saisies si une adresse est spécifiée.** Les mots en ⟨emphases⟩ doivent être remplacés par les valeurs voulues.

Exercice 1 : Utilisation de base de `ed`

Q1. Connectez-vous en mode graphique, ouvrez un émulateur de terminal puis assurez-vous que votre dossier de travail soit le dossier `R1.03/TP02` (créez le si nécessaire).

Q2. Lisez[4] complètement la page info de `ed` en passant les sections `Invoking Ed`, `Regular Expressions` et toutes celles venant après `Commands`.

Q3. Créez maintenant un fichier nommé `information` contenant 3 lignes puis quittez `ed` :

1. La première ligne du fichier doit contenir votre nom,
2. la seconde votre prénom,
3. la dernière votre groupe.

Q4. Depuis le shell vérifiez que le contenu de votre fichier correspond bien à ce que vous avez saisi grâce à la commande `cat`.

Q5. Ouvrez le fichier `information` avec `ed` et faites afficher le contenu du fichier avec une numérotation des lignes.

Exercice 2 : Édition d'un long texte avec `ed`

Q1. Démarrez l'éditeur `ed`.

Q2. Insérez le texte suivant, **en respectant les longueurs des lignes** et **en ne saisissant pas les numéros de lignes** :

4. Ici, lire veut dire lire **et** comprendre

1905

```
Extraits de la loi de 1905 concernant la séparation des Églises et de l'État
================================================================================

- **Article premier.** La République assure la liberté de conscience. Elle
  garantit le libre exercice des cultes sous les seules restrictions
  édictées ci-après dans l'intérêt de l'ordre public.

Extraits d'un éditorial du dessinateur Siné
===========================================

> Comme sur les paquets de clopes où figure maintenant en gros *FUMER
> TUE*, on devrait inscrire, en énorme, sur les couvertures de la bible,
> de la thora et du coran : **LA RELIGION REND CON**.
```

Q3. Sans quitter `ed`, vérifiez en le faisant afficher que le texte que vous avez saisi est bien en mémoire.

Q4. Sans quitter `ed`, sauvegardez le contenu dans un fichier de nom `1905.md`.

Q5. Sans quitter `ed`, insérez le contenu du fichier `/home/public/baste/R1.03/TP02/1905`, **après** la ligne 6.

Q6. Sans quitter `ed`, déplacez les extraits de l'éditorial de Siné (de la ligne 18 à la dernière ligne) en début de fichier.

Q7. Sans quitter `ed`, insérez le texte suivant entre l'avant dernière et la dernière ligne :

1905-ajout

```
> L'anarchiste est celui qui a un tel besoin d'ordre qu'il n'en admet aucune parodie. Antonin Artaud.
```

Q8. Sans quitter `ed`, vérifiez que toutes les modifications que vous avez faites sont bien en mémoire en faisant afficher le contenu de la mémoire.

Q9. Quitter `ed` en enregistrant vos modifications.

L'éditeur de texte visuel `vi`

Contrairement aux outils en mode ligne qui n'utilisent que le minimum des capacités offertes par les terminaux (la ligne), `vi` est dit *visuel* car il représente le contenu d'un fichier texte en utilisant toute la zone d'affichage offerte par certains terminaux, comme les écrans, dans lequel il est utilisé. Une partie de la mémoire, correspondant au contenu du fichier, est donc constamment visible à l'écran. La partie visible dépend de la taille de l'écran du terminal.

Comme `ed`, `vi` est un éditeur de texte fonctionnant grâce à des modes. Il existe deux modes d'utilisation sous `vi` :

1. le mode **commande**[5] : au démarrage de `vi` vous êtes dans ce mode. Une commande c'est un ou plusieurs caractères qui permettent de demander un traitement particulier (déplacement, recherche, remplacement, etc.). Il y a 2 types de commandes :
   — les *commandes vi* sont utilisées en appuyant sur la (ou les) touche(s) leur correspondant
   — les *commandes ed* sont données sur la dernière ligne de votre terminal. Pour cela vous devez :
   (a) utiliser la commande `vi :`. Le curseur se positionne alors sur la dernière ligne de votre terminal.
   (b) saisir votre commande `ed`.
   (c) valider la requête pour la faire s'exécuter en appuyant sur la touche [entrée].
2. le mode **saisie** : il existe plusieurs commandes `vi` permettant de passer dans ce mode. Le seul moyen d'en sortir (de s'en *échapper*) est d'appuyer sur la touche d'échappement : [échap]. Comme son nom l'indique ce mode permet de saisir du texte. La différence avec `ed` est que le texte saisi apparaît directement à l'écran à l'endroit où il est ajouté.

La *position d'édition*, ou *position courante*, dans le fichier est repérée par le *curseur* (un caractère représentant un bloc noir ou un trait clignotant). Le caractère se trouvant sous le curseur est appelé le *caractère courant*, il correspond à un caractère du fichier. Pour déplacer le curseur et donc la position d'édition, il existe 4 commandes `vi` particulières :

1. `h` déplace le curseur d'un caractère vers la gauche
2. `l` déplace le curseur d'un caractère vers la droite
3. `j` déplace le curseur d'une ligne vers le bas
4. `k` déplace le curseur d'une ligne vers le haut

Ces commandes `vi` fonctionnent quelque soit le terminal utilisé. Cependant sur les machines de TP vous pouvez de manière plus simple utiliser les flèches de votre clavier pour effectuer ces déplacements, respectivement [←], [→], [↓] et [↑].

Dans les descriptions de commandes et exemples du reste de ce TP les mots en ⟨emphase⟩ représentent des informations que vous devez fournir.

— Pour démarrer l'édition d'un fichier il suffit de taper la commande `vi fichier`.

5. On dira aussi **directive**

— Pour sauvegarder un fichier une fois dans `vi` il vous faut utiliser la commande `:w`
— Pour quitter l'éditeur :
  — si vous n'avez pas modifié le fichier il suffit d'utiliser la requête `:q`
  — si vous avez modifié le fichier mais que vous ne désiriez pas sauvegarder les modifications il vous faut utiliser la commande `:q!`
  — si vous avez modifié le fichier et que vous voulez sauvegarder ces modifications il vous faut utiliser l'une des requêtes `:x`, `:wq` ou simplement `ZZ`.

Le tableau 2 rassemble un certain nombre des commandes `vi` importantes. La page du manuel de `vi` vous en dira plus.

Il est possible de répéter l'éxécution d'une commande `vi` en la préfixant par le nombre de fois qu'elle doit être éxécutée. Par exemple pour effacer 10 lignes il suffit de taper `10dd`.

Lorsqu'il vous semble que vous ne vous souvenez pas dans quelle mode vous êtes et que vous êtes perdu un bon réflexe est d'appuyer plusieurs fois sur la touche échap jusqu'à ce que vous entendiez un bip sonore signalant que vous êtes de nouveau en mode commande (`vi` ne connaissant pas de directive liée à la touche échap vous signale une erreur en émettant un bip sonore).

Commandes `vi`	
`h`	Déplacer le curseur d'un caractère vers la gauche
`l`	Déplacer le curseur d'un caractère vers la droite
`j`	Déplacer le curseur d'une ligne vers le bas
`k`	Déplacer le curseur d'une ligne vers le haut
`0`	Positionner le curseur en début de ligne
`$`	Positionner le curseur en fin de ligne
`:0`	Positionner le curseur sur la première ligne du fichier
`:$`	Positionner le curseur sur la dernière ligne du fichier
`u`	Annuler la dernière commande
`.`	Répéter la dernière commande
`i`	Passer en mode de saisie (insertion avant le curseur)
`I`	Passer en mode de saisie (insertion avant la ligne courante)
`a`	Passer en mode de saisie (ajout après le curseur)
`A`	Passer en mode de saisie (ajout après la ligne courante)
`yy`	Copier la ligne courante dans un tampon
`P`	Coller le contenu du tampon avant la ligne courante
`p`	Coller le contenu du tampon après la ligne courante
`x`	Supprimer le caractère courant
`dw`	Supprimer le mot courant
`dd`	Supprimer la ligne courante
`ZZ`	Quitter après avoir sauvegarder les modifications
`/`⟨motif⟩	Rechercher ⟨chaine⟩
`n`	Répéter la dernière recherche effectué à partir de la position courante
`:`⟨commande⟩	Effectuer la ⟨commande ed⟩ spécifiée
`:set` ⟨option⟩	Fixer l'⟨option⟩ de l'éditeur (cf table 3)
`:set no`⟨option⟩	Enlever l'⟨option⟩ de l'éditeur (cf table 3)
`:syntax on`	Activer la colorisation syntaxique

TABLE 2 – **Commandes `vi`**. Les mots en ⟨emphases⟩ doivent être remplacés par les valeurs voulues. La plupart des commandes peuvent être préfixées par le nombre de fois qu'elles doivent être exécutées.

Options de `vi`	
`number`	Afficher les numéros de lignes devant chaque ligne.
`ignorecase`	Ne pas faire de distinction entre les majuscules et les miniscules pour les recherches.
`autoindent`	Indenter automatiquement les programmes en fonction du langage de programmation utilisée.

TABLE 3 – **Options de l'éditeur `vi`**

Exercice 3 : Édition avec `vi`

Q1. Ouvrez votre fichier en utilisant `vi`.

Q2. Insérez vos noms et prénoms en tête du fichier sur une seule ligne.

Q3. Copiez l'article 26 en début de fichier 20 fois.

Q4. Utilisez la commande de substitution pour remplacer partout dans le texte `culte` par `cuculte`.

Q5. Ajoutez en fin de fichier le contenu du fichier `information` que vous avez créé dans le premier exercice.

Q6. Utilisez la commande de substitution pour remplacer partout dans le texte `religi` par `/piege/ /a/ /`.

Q7. Comptez en utilisant les commandes de recherche de chaîne le nombre de lignes contenant une ou plusieurs occurrence du mot `les`.

Q8. Appelez la commande `man vi` sans quitter `vi`.

Q9. Quittez l'éditeur en sauvegardant le contenu.

Exercice 4 : Apprendre à utiliser Emacs (Bonus)

Q1. Démarrer l'éditeur Emacs et faites en sorte que sa fenêtre soit maximisée sur votre écran.

Q2. Appuyer simultanément sur les touches Ctrl et h puis sur la touche t .
Un tutoriel vous est proposé suivez-le **entièrement** (*i.e.* jusqu'à la fin) **avant la prochaine séance**.

Exercice 5 : Apprendre à utiliser `vim` (Bonus)

L'implémentation de `vi` installée sur vos postes se nomme `vim`. La commande `vimtutor` vous permet de suivre un tutoriel pour apprendre à s'en servir correctement.

Q1. Suivez ce tutoriel **entièrement** (*i.e.* jusqu'à la fin).