Projet : Editeur de texte

Dernière modification : 11 mars 2019

{pascal.vanier}@u-pec.fr

Attention ! Le sujet peut être mis à jour dans les prochaines semaines, pensez à le reconsulter de temps à autre.

Il faut lire le sujet intégralement avant de commencer le projet [1].

Consignes

Le projet devra être codé intégralement en C, tout ajout non demandé ne sera pris en compte dans la note que si les fonctionnalités de base sont implémentées. Le projet est à réaliser **seul ou à deux**.

Les ouvertures/lectures/écritures de fichiers devront être faites avec les appels systèmes, vous avez cependant le droit d'utiliser les fonctions de `<string.h>` ainsi que les fonctions `sprintf`, `snprintf` et `perror`. Il devra comporter un `Makefile` et il devra compiler avec les options `-std=gnu99 -Wall -Werror`.

Vous joindrez à votre projet une documentation expliquant son fonctionnement, ses limites, les problèmes que vous avez rencontrés et les solutions que vous avez trouvées et les contributions des divers membres du groupe. Celle-ci devra se trouver dans un fichier `readme.md` se trouvant à la racine de votre dépôt [2].

La notation tiendra compte de divers paramètres : la qualité du code (lisibilité, modularité, commentaires), la qualité des structures de données, la qualité algorithmique, le fonctionnement du programme (phase réussie, présence ou absence de bugs), ainsi que la soutenance.

Il est obligatoire d'utiliser le gestionnaire de versions `git`, un compte sera créé pour chacun d'entre vous sur le serveur https://git-etudiants.lacl.fr dès que les groupes auront été formés, avec ces comptes seront créés des dépôts pour chacun des groupes. Vos `Makefile` et `README.md` doivent se trouver à la source du répertoire.

Attention : Tout projet qui ne sera pas rendu sur `git` ne sera pas noté. Vous devez vous en servir *régulièrement* [3]. Par ailleurs, seul le code source doit se trouver sur git, pas les exécutables. Tout projet rendu sous forme de fichier compressé sur git ne vaudra aucun point.

Dates

Les groupes sont à former avant le **28 février 2019**, sur eprel. Si vous souhaitez faire le projet seul, inscrivez-vous sur un groupe ne pouvant accueillir qu'une seule personne pour éviter que quelqu'un ne se rajoute.

La version finale du projet doit être sur `git` avant le **18 mai 2019**. La soutenance aura lieu le **21 mai 2019**.

1. Oui, je sais, cela paraît évident...
2. Les fichiers `md` sont des fichiers markdown, une documentation est disponible sur https://docs.gitlab.com/ce/user/markdown.html.
3. Cela signifie en particulier que les excuses comme "j'ai perdu mes fichiers par accident", "nous avons rajouté des bugs au dernier moment" ne seront pas acceptées.

Le principe

Le but de ce projet est de faire un éditeur de texte : un clone de *vim* [4]. Il s'agit d'un éditeur de texte dans le terminal comprenant trois modes :

— Un **mode insertion** dans lequel on peut éditer du texte normalement : le texte tapé doit s'afficher là où est le curseur, on peut se déplacer dans le texte avec les flèches, on peut supprimer des caractères avec SUPPR ou BACKSPACE.

— un **mode normal** dans lequel on peut taper des instructions qui consistent en des déplacements ou changements de mode : aller au mot suivant, à la lettre suivante, à la ligne suivante, insérer une ligne après/avant la ligne courante et passer en mode insertion, etc... On peut passer au mode normal à partir du mode insertion en tapant sur ESC, on peut passer du mode normal au mode insertion à l'aide de `i`.

— Un mode visuel dans lequel on peut selectionner du texte intelligemment, on peut ensuite modifier/supprimer ce texte avec des commandes. On ne s'intéressera pas à ce mode (sauf pour les éventuels bonus).

Attention : le programme ne devra être constitué que d'un seul thread.

1 Phase 1 : un éditeur de texte simple – mode insertion avec clavier

1.1 Gestion du terminal

La première étape sera de vous familiariser avec `termios`, allez faire un tour sur la page `man termios`, afin d'être capables d'afficher une "image" ascii dans votre terminal. On souhaite également qu'à cette étape les entrées au clavier ne soient pas nécessairement affichées (le terminal ne devra donc pas rester en mode canonique).

La première étape du projet sera donc d'être capable d'afficher un fichier ouvert avec votre éditeur en ligne de commande (le premier argument donnée en ligne de commande pourra être un fichier que l'on ouvrira et affichera). On prendra bien garde à n'afficher que le début du fichier, c'est à dire uniquement les caractères qui tiennent à l'écran.

1.2 Gestion de l'édition simple

Dans un second temps vous ferez en sorte que l'on puisse modifier le texte : il faudra donc que vous trouviez un moyen d'afficher un curseur dans le terminal (par exemple en changeant la couleur d'arrière plan du caractère où se trouve le curseur).

Vous implémenterez également un mode normal basique :

— les déplacements avec les flèches fonctionneront,

— vous pourrez passer du mode insertion au mode normal avec ESC,

— vous pourrez passer du mode normal au mode insertion avec `i`,

— vous implémenterez l'invite de commandes complexes qui commencera quand on tape `:` et s'arrête soit quand on tape ENTER soit quand on efface toute la commande (inspirez vous du véritable vim). Vous implémenterez les trois commandes `:q`, `:w` et `:w fichier`.

2 Phase 2 : un éditeur de texte simple – gestion de la souris

Vous implémenterez la gestion de la souris en vous servant de `/dev/input/mice`, qui est un fichier spécial dans lequel on peut lire les déplacements de la souris. Pour pouvoir lire ce

4. Vous pouvez l'installer dans n'importe quelle distribution pour voir son fonctionnement.

fichier il faut ajouter l'utilisateur courant à un groupe, à vous de configurer votre machine pour pouvoir le lire (on a vu en cours quel fichier modifier pour cela).

Les déplacements lus sur `/dev/input/mice` sont relatifs, c'est à dire qu'il s'agit de coordonnées par rapport à la position courante du curseur. Vous implémenterez un second curseur [5] dans votre terminal qui correspondra à la position de la souris (il n'est pas nécessaire que la position de votre curseur corresponde avec la position du pointeur de la souris dans l'interface graphique).

Il faut que les deux modes de déplacement fonctionnent en même temps.

3 Phase 3 : améliorations

Vous implémenterez une ou plusieurs améliorations, la note maximale ne pourra être atteinte que si plusieurs améliorations non triviales ont été implémentées. Voici quelques idées, vous pouvez également implémenter vos propres idées :

— (*) recherche de texte avec / en mode normal. On pourra éventuellement faire cela efficacement avec l'algorithme de Knuth-Morris-Pratt (plus difficile).
— (**) remplacement de texte simple.
— (*) buffers multiples (voir :b dans vim).
— (**) Mode visuel (avec éventuellement sélection du texte à la souris)
— ...

5. Vous pourrez utiliser une couleur différente que pour le curseur principal.

A Manipuler le terminal

A.1 Taille du terminal

Votre programme doit fonctionner pour une taille fixée de terminal que vous pourrez choisir. Ne choisissez pas une taille trop grande, Un bon moyen d'imposer la taille d'un terminal est de lancer votre programme avec la commande suivante :

```
xterm -geometry 80x24 -e /chemin/vers/programme
```

vous pouvez remplacer 80 et 24 par les valeurs que vous souhaitez. Il est également possible d'accèder à la hauteur/largeur d'un terminal en accédant aux variables d'environnement `LINES` et `COLUMNS`.

A.2 Mode du terminal

Le mode par défaut dans lequel est le terminal est le mode canonique, c'est à dire qu'un read débloque uniquement quand l'utilisateur tape sur la touche entrée et tout ce qui est entré au clavier est affiché à l'écran. Il faut, afin que le terminal n'affiche pas les touches utilisées et permette de lire les touches sans que l'on utilise entrée, passer le terminal en mode `raw`, on peut pour cela faire appel à la fonction `tcgetattr` afin de récupérer les attributs du terminal, modifier ces attributs avec `cfmakeraw`, puis les attribuer au terminal avec `tcsetattr`. Le `man termios` ainsi que de chacune de ces fonctions est votre ami.

A.3 De la couleur dans le terminal

On peut utiliser de la couleur dans un terminal, afin de faire cela, on peut écrire avec `write` le résultat de `sprintf(buffer, "\x1b[91m%s", acolorier);`. Après cela, toutes les autres chaînes qui seront écrites seront de cette couleur, il faut donc remettre la couleur initiale si l'on ne veut pas cela.

Vous pouvez consulter la page http://jafrog.com/2013/11/23/colors-in-terminal.html pour plus d'informations sur les couleurs et autres attributs du texte dans le terminal (lien en anglais).

A.4 Se déplacer dans le terminal

Vous pouvez déplacer le curseur à la position (x, y) en écrivant le résultat de

```
sprintf(buffer, "\x1b[%d;%dH", x, y);
```

avec `write`.

A.5 Cacher le curseur

Pour cacher le curseur, il suffit d'utiliser le code `\x1b[?251` et pour le réafficher `\x1b[?25h`.