L'une des forces de beaucoup de programmes du monde Unix est la facilité à les configurer pour les modeler à l'usage que l'on veut en faire. L'objectif de ce TP est de comprendre comment on peut configurer un programme.

Ce TP est un TP lié à la SAÉ 1.03. À ce titre, ce qui est demandé dans ce TP est susceptible d'être aussi demandé au CTP de S1.03. Il n'est ici pas attendu que vous connaissiez par coeur toutes les configurations possibles de `nano`, mais que vous soyez capable de retrouver rapidement une configuration particulière si on vous la demande. Pour vous donner une idée, ce que vous faites ici en 3h de TP, car il vous faut le temps de comprendre le fonctionnement et les subtilités des configurations, correspondra à une ou plusieurs questions qui devront vous mettre au plus 5 min à répondre pendant l'évaluation.

Si, comme nous allons voir, configurer un programme est quelque chose de relativement simple, le faire correctement nécessite une lecture approfondie. Il y aura donc pas mal de lecture pendant ce TP. En effet avant de personnaliser un programme, il faut en premier lieu savoir quel est son comportement par défaut puis ensuite comprendre comment le programme peut être personnalisé. Pour cela, il n'y a pas de mystère, la documentation est la seule source complète d'information.

1 Nano

Dans ce TP, nous allons personnaliser l'éditeur de texte `nano`. Cet éditeur de texte, dont vous vous êtes déjà servi dans le TP01 peut paraitre austère à première vue, mais devient beaucoup plus intéressant une fois pris en main.

Exercice 1.1 : Comprendre `nano`

Comme dit précédemment, avant de se lancer dans la personnalisation de `nano`, il faut en premier lieu savoir ce qui peut être fait avec sans rien toucher.

Q1. Lisez les sections NAME, DESCRIPTION et EDITING du manuel de `nano`.

Q2. Testez chacune des options et des raccourcis clavier qui vous sont expliqués dans les sections DESCRIPTION et EDITING.

Q3. Faite de même avec la section TOGGLES.

Q4. En relisant les sections DESCRIPTION, EDITING et TOGGLES si nécessaire, trouvez la liste de tous les raccourcis clavier de nano

Q5. À la fin de cet exercice, vous devez au moins être capable de :

1. Sélectionner une zone de texte à partir du clavier
2. Copier cette zone de texte dans le presse papier
3. Coller depuis le presse papier
4. Annuler votre dernière action
5. Commenter/décommenter une ligne
6. Chercher un mot dans le document
7. Effectuer un chercher/remplacer dans le document
8. Vous déplacer rapidement vers une ligne précise du document
9. Centrer nano sur la ligne où se trouve votre curseur
10. Compter le nombre de caractères
11. Enregistrer et exécuter une macro
12. Placer des ancres dans le document pour vous déplacer rapidement
13. Afficher le numéro des lignes

Exercice 1.2 : Votre première configuration

Maintenant que vous savez utiliser `nano`, il est temps de commencer votre première configuration. La lecture de la section FILE du manuel de `nano` vous informe que `nano` va lire le fichier `~/.nanorc` (s'il existe) pour déterminer la configuration qu'il doit appliquer. Notez que le fichier `~/.nanorc` est un fichier qui est dans votre home, de ce fait, la configuration que vous y écrivez vous sera propre. Notez aussi que le fichier `~/.nanorc` commence par un "`.`", c'est donc un fichier caché. Pour le voir, il faudra utiliser l'option `-a` lorsque vous utiliserez la commande `ls`.

Q1. Créez le fichier `~/.nanorc` et ajoutez-y la ligne `set linenumbers`. Enregistrez votre fichier et relancez `nano`, que constatez-vous ?

Vous venez d'effectuer votre première configuration. Grâce à cela, le comportement usuel de `nano` à changé pour répondre à la nouvelle attente que vous avez de lui. L'affichage du numéro des lignes ici est symbolique, mais peut se révélé très utile surtout puisque comme on l'a vu dans l'exercice précédent, il est possible avec nano de se déplacer directement à une ligne précise. De plus, notez que les ancres que vous pouvez disposer dans votre document ont elles aussi un visuel.

Maintenant que vous êtes lancé, il serait triste de se limiter à une seule configuration. Vous pouvez compléter votre fichier de configuration à raison d'une configuration par ligne. Pour connaitre toutes les configurations possibles, le manuel est à nouveau votre ami.

Q2. Parcourez le manuel de `nanorc` et en particulier la section OPTION. Ajoutez les configurations qui vous parlent. N'oubliez pas de redémarrer `nano` pour tester vos nouvelles configurations. Vous devez au moins changer les couleurs de la barre de titre ainsi que des lignes d'aide présentes en bas de `nano`. Cherchez aussi comment ajouter un indicateur de votre positionnement dans le document.

Exercice 1.3 : Modification des raccourcis clavier

Comme dit précédemment `nano` (comme beaucoup de programmes GNU) est hautement configurable. En particulier, s'il existe des raccourcis par défaut à `nano`, rien ne vous empêche de les changer, les supprimer ou en rajouter d'autres. La section REBINDING KEYS du manuel de `nanorc` explique comment effectuer ces changements.

Q1. Lisez l'entête de la section REBINDING KEYS.

Q2. Supprimer le raccourci clavier `^K` permettant de couper la ligne sur laquelle vous êtes. Ici le menu à considérer est `all`. Relancez `nano` et constatez ce qu'il en est.

Q3. Faites en sorte que la touche `F12` vous permette de commenter ou de décommenter une ligne (comme le fait déjà la combinaison `M-3`).

Q4. Ajouter un raccourci clavier sous `^K` qui vous permet d'insérer le texte "`truc`" au niveau de votre curseur.

Q5. Prenez le temps de modifier les raccourcis clavier que vous voulez.

Exercice 1.4 : Modification simple de la coloration syntaxique

Il est possible avec nano d'avoir de la coloration syntaxique pour un fichier spécifique. La section SYNTAX HIGHLIGHTING du manuel de `nanorc` explique comment mettre en place cette coloration.

Q1. Dans un premier temps, on ne va s'intéresser qu'aux mots clefs `syntax` et `color`. Créez une syntaxe `test` qui colore toutes les occurrences du mot `set` en vert (ou autre couleur). Aide : Dans l'utilisation de `color`, le mot `"regex"` doit être remplacé par le mot clef dont vous voulez changer la couleur.

Q2. Faites en sorte que les mots clefs `bind`, `unbind`, `syntax` et `color` soit aussi en vert. Assurez-vous que tout cela est fait une une seule ligne de configuration.

Q3. Cherchez le mot clef qui permet de modifier le ou les caractères marquant la présence d'un commentaire. Dans notre cas, on mettra cette valeur à `#`. Testez ce qu'il se passe si vous changez cette valeur puis utilisez la touche `F12` que vous avez configurée précédemment.

Q4. Faites en sorte que les chaînes de caractères écrites entre guillemets (`""`) apparaissent en bleu.

Exercice 1.5 : Modification de la coloration syntaxique par regexp

Les expressions régulières (aussi appelées regexp) permettent au lieu de donner un mot précis, de donner la forme d'un mot. Par exemple la regexp `t.t.` va correspondre à n'importe quel mot qui commence par un `t` puis un caractère quelconque puis un `t` puis un autre caractère quelconque. Par conséquent, lors de la recherche d'un mot de la forme `t.t.`, le programme va s'arrêter lorsqu'il voit le mot `toto`, mais aussi `tata`, `tutu` ou autres dérivés. Notez que le caractère quelconque représenté par le `.` n'est pas forcément deux fois le même, ainsi le programme s'arrêtera aussi sur les mots `tate`, `tito` ou autres.

Q1. Pour la configuration de `nano`, les lignes commençant par `#` sont des lignes commentées. En utilisant une regexp à base d'un `#` suivit de deux "`.`", faites en sorte que pour chaque ligne commentée, le caractère `#` et les deux caractères suivants soient écrits en rouge.

Q2. Les regexp permettent de faire des choses bien plus jolies que ce qui est fait à la question précédente. Lisez avec attention le deuxième paragraphe de la section SYNTAX HIGHLIGHTING et faites en sorte que lorsqu'une ligne est commentée, toute la ligne soit en rouge.

Q3. Dans l'exercice précédent, vous avez fait passer le mot clef `color` en vert. En réalité vous ne vouliez colorer que les mots clefs qui sont au début d'une ligne. À l'aide d'une expression régulière, corriger cette coloration syntaxique.

Q4. Faites en sorte que lorsque vous avez une virgule, le mot avant et le mot après soit écrit en magenta. Aide : À la place du "`.`", vous pouvez utiliser `[a-z]` pour forcer que le caractère soit une lettre de l'alphabet écrit en minuscule.

Exercice 1.6 : Finitions

Q1. Maintenant que vous avez manipulé la configuration de nano un peu dans tous les sens, épurez votre fichier en configuration (en commentant plutôt qu'en effaçant) afin d'obtenir une configuration qui vous corresponde.

Q2. Profitez de vos nouvelles compétences pour écrire une syntaxe pour vos codes en ijava. (On ne sait jamais, des fois que ça vous donne envie de faire vos TP de ijava avec nano.)

Q3. La section FILES du manuel de `nanorc` vous indique que des syntaxes pour les langages les plus courants sont déjà écrit et présent dans le dossier `/usr/share/nano/`. Observez le fichier `java.nanorc` qui peut vous donner de l'inspiration pour votre syntaxe de ijava.

2 Bash

Maintenant que nous avons configuré `nano`, pourquoi ne pas aussi configurer votre environnement `bash` ? Si vous êtes arrivé jusqu'ici, c'est que vous êtes plus en avance que prévu. Cette section comprendra uniquement le minimum nécessaire pour comprendre sans faire de bêtises, pour le reste, ça sera à vous de vous lancer.

Exercice 2.1 : Évitons de tout casser

Dans le cas de `nano`, le fichier de configuration était `~/.nanorc` et il n'existait pas au début du TP. Dans le cas de `bash`, c'est différent, vous avez déjà un fichier `~/.bashrc` présent sur votre machine. Comme, lorsque vous faites des modifications, vous risquez de détruire la configuration présente précédemment, il est important de sauvegarder la configuration que vous avez actuellement pour pouvoir revenir à celle-ci en cas de besoin.

Q1. Copiez le fichier `~/.bashrc` dans le dossier `~/R1.04/TP02` que vous aurez créé pour l'occasion.

> Dans le cadre des fichiers `~/.bashrc` présents sur les machines du département, il est indispensable que vous laissiez inchangées les deux premières lignes. Vous devez donc écrire votre configuration à la suite à partir de la troisième ligne.

Faire une sauvegarde d'un fichier de configuration fonctionnel est une habitude importante à prendre. Certains programmes ont besoin d'une configuration précise pour fonctionner sur une machine précise et si vous perdez la configuration, le programme ne peut plus s'exécuter correctement. Il ne vous restera alors plus qu'à relire toute la documentation du programme pour reconstruire la configuration nécessaire (indice, ça peut prendre pas mal de temps).

Exercice 2.2 : Comment fonctionne le bashrc

Chaque programme à une manière d'être configuré qui lui est propre. Ainsi, pour `nano`, le manuel vous a donné une syntaxe à respecter qui est propre à la configuration de `nano`. Dans le cas de `bash`, c'est différent. Le fichier `~/.bashrc` n'est pas un fichier de configuration à proprement parlé, mais un ensemble de commandes qui sont exécutées à chaque lancement de `bash`. La syntaxe du fichier de configuration est donc la même que la syntaxe des commandes que vous pouvez passer en utilisant `bash`.

Q1. La commande `date(1)` affiche la date et l'heure. Ajoutez `date` sur une nouvelle ligne dans votre fichier `~/.bashrc`. Ouvrez un nouveau terminal. Que constatez-vous ? Dans ce terminal exécutez la commande `bash`, que se passe-t-il ?

Q2. Comme dit précédemment, pour configurer `bash`, il faut invoquer des commandes `bash`. Pour savoir tout ce que vous pouvez faire, il faut donc aller lire le manuel de `bash`. Le manuel est très long, il va donc falloir que vous vous concentriez sur des points précis.

Q3. Parmi les choses simples à modifier, il y a la variable `$PS1` qui est la variable qui explique ce que doit afficher le prompt. Ainsi l'ajout de la ligne `PS1=toto$` changera votre prompt en `toto$`. Astuce : La lecture de la section PROMPTING vous aidera à personnaliser votre prompt.

Q4. La notion d'alias permet de donner un nom court à une commande que vous effectuez régulièrement. Trouvez comment créer un alias.

Q5. Votre fichier `~/.bashrc` précharge une configuration définie par les administrateurs. Allez consulter le fichier en question pour connaitre la configuration réelle de votre session `bash`.

Exercice 2.3 : À vous de jouer !

Personnalisez votre bash comme vous le désirez.