Exercice 1 : Manipuler les raccourcis claviers sous linux et quelques commandes linux pratiques

Bien organiser son espace de travail, c'est se simplifier la vie par la suite.
Il existe des astuces pratiques pour organiser son espace de travail comme celle que nous allons voir pour positionner ses fenêtres sur un quart d'écran / demi-écran.

Sélectionner une fenêtre, cliquer sur la barre du haut de la fenêtre avec le clic droit, garder enfoncer le clic droit et déplacer dans les coins la fenêtre.
Que se passe-t-il ?

Pour la suite du TP, organiser votre espace de travail de la façon suivante :

* un terminal sur un quart de l'écran
* le sujet du tp sur une demi-partie de l'écran.

Lancer un terminal avec le raccourci : ^ALT+T
Pour fermer un terminal : ^D

Lorsqu'un terminal est ouvert, il est très important de se souvenir de certaines commandes.
Parmi ces commandes à ne jamais oublier quand on utilise un terminal :
^C : arrête le processus en cours
^S : gèle l'affichage de l'écran =======> ^Q : reprendre l'affichage normal

Exécuter la commande : cat /dev/urandom
Que se passe-til ?

Vous avez perdu la main sur le terminal.

Tester les raccourcis pour geler l'affichage de l'écran, puis reprendre l'affichage normal.
Enfin, stopper ce processus.

La deuxième partie de cet exercice présente une manière d'afficher le contenu d'un fichier.
Les plus utilisées sont less (more), et cat.

Exécuter les commandes suivantes et expliquer ce qu'elles font :
cat /usr/share/dict/words

cat /usr/share/dict/words | head -10

cat /usr/share/dict/words | tail -10

less /usr/share/dict/words
Pour quitter less, il suffit d'appuyer sur la lettre "q".
"h" pour obtenir le manuel d'aide.

Pour effectuer une recherche par exemple, il suffit de taper le caractère "/".
Effectuer la recherche "Ab".
Pour parcourir les éléments suivants "n", pour revenir aux précédents "N".

Exercice 2: Utiliser un éditeur de fichier dans un terminal

Lorsque nous utilisons des terminaux, il est souvent nécessaire d'écrire ou d'éditer des fichiers textes depuis le terminal.
Il existe plusieurs éditeurs de textes sous lignes de commandes : jed, nano, vim, ...
Remarquons, qu'il est important de bien distinguer les commandes pour afficher un fichier (cat, less), des commandes pour éditer d'un fichier(jed, nano, vim).
Ainsi, sous linux, il n'est pas commode d'utiliser une commande comme nano pour afficher le contenu d'un fichier.

L'objectif de cet exercice est de découvrir l'un des plus simples à utiliser : nano.

1. Ouvrir un terminal et exécuter la commande :
   nano

2. Compléter :

   CTRL + G : Affiche l'aide

   ? : Permet de sauvegarder votre fichier

   ? : Permet de quitter nano

   ? : Permet de faire une recherche

D'autres commandes :
   CTRL + A : ?

   CTRL + E : ?

   CTRL + Y : ?

   CTRL + V : ?

   CTRL + _ : ?

   CTRL + C : ?

   CTRL + D : ?

   CTRL + K : ?

3. Quitter maintenant nano !

Il existe différentes options d'utilisation de nano.

Essayer différentes options et préciser ce que font ces options :
nano -l

nano -i

nano -m

4. a) Utiliser le raccourci clavier CTRL + SHIFT + T pour ouvrir un nouvel onglet dans votre écran de terminal, pour le fermer, vous pouvez utiliser le raccourci CTRL + SHIFT + D pour fermer un onglet ou le terminal.

b) Dans le nouvel écran, créer le fichier texte avec la commande touch et nommer le “code.java”.

c) Afficher le fichier texte dans le nouvel écran de terminal avec la commande less ou cat.

```
//	private PropositionalFormula getOneElement() {
//		Iterator it = (Iterator) this.iterator();
//		if(it.hasNext()) {
//			PropositionalFormula prop =  ((java.util.Iterator<Set<PropositionalFormula>>) it).next();
////		return prop;
//		} else {
//			return null;
//		}
//	}
```

Pour copier en SHELL ou nano : CTRL + SHIFT + C
Pour coller en nano : CTRL + SHIFT + V

d) Enregistrer ce fichier sous le nom "code.java"
Pour info, une autre manière pratique pour coller consiste à simplement sélectionner un bout de texte (laisser surligné) puis de cliquer avec la molette de la souris à l’emplacement souhaité.

e) Ouvrir nano dans l'autre écran et copier/coller le texte affiché du premier écran de terminal dans l'éditeur nano !

5. Ces options d'affichage particulières pour nano peuvent être permanentes dans le lancement par défaut de nano. Pour ce faire, il existe un fichier de configuration de nano dans le dossier :

/etc/nanorc

Ce fichier correspond à la configuration par défaut pour tous les utilisateurs de la machine.

a) Afficher le contenu de ce fichier
Essayer d'éditer ce fichier avec nano. Que se passe-t-il ?

b) A quoi correspond le dossier ~/ ?

c) Copier coller /etc/nanorc dans le dossier ~ sous le nom .nanorc avec la commande destinée à faire ça dans un terminal !

d) Pourquoi le "." devant le nom de fichier ?

e) Une fois le fichier collé, éditer les lignes en ouvrant avec nano le fichier collé ~/.nanorc:

set autoindent
...
set backup
...
include …
...
set linenumbers
...
set mouse
...
set matchbrackets "(<[{)>]}"

Supprimer les "#" devant.

f) Enregistrer les modifications avec nano et relancer nano. Que s'est-il passé ? A quoi servaient ces "#" ? Comment appelle-t-on ces lignes "inutiles" dans le jargon de l'informatique ?
A quoi correspondaient ces lignes ?

g) nano peut faire encore plus pour vous, vous pouvez personnaliser la coloration du texte.
(source : https://blog.shevarezo.fr/post/2018/01/11/ajouter-coloration-syntaxique-nano-linux)
Nous allons d'abord afficher la coloration des fichiers de code écrits en C, et pour ce faire :

g1) Créer un dossier ~/.nano/ avec la bonne commande linux (mkdir)

g2) Editer le fichier ~/.nanorc et ajouter la ligne:

g3) include "~/.nano/c.nanorc"

g4) Que fait cette ligne ajoutée ??

g5) Créer un fichier dans le dossier ~/.nano/c.nanorc, coller le fichier suivant, puis enregistrer ces modifications :

```
syntax "C" "\.(c(c|pp|xx)?|C)$" "\.(h(h|pp|xx)?|H)$" "\.ii?$" "\.(def)$"
color brightred "\<[A-Z_][0-9A-Z_]+\>"
color green "\<(float|double|bool|char|wchar_t|int|short|long|sizeof|enum|void|static|const|struct|union|
typedef|extern|(un)?signed|inline)\>"
color green "\<((s?size)|(char(16|32))|((u_?)?int(_fast|_least)?(8|16|32|64))|u?int(max|ptr))_t\>"
color green "\<(class|namespace|template|public|protected|private|typename|this|friend|virtual|using|
mutable|volatile|register|explicit)\>"
color green "\<(for|if|while|do|else|case|default|switch)\>"
color green "\<(try|throw|catch|operator|new|delete)\>"
color green "\<((const|dynamic|reinterpret|static)_cast)\>"
color green "\<(alignas|alignof|asm|auto|compl|concept|constexpr|decltype|export|noexcept|nullptr|
requires|static_assert|thread_local|typeid|override|final)\>"
color green "\<(and|and_eq|bitand|bitor|not|not_eq|or|or_eq|xor|xor_eq)\>"
color brightmagenta "\<(goto|continue|break|return)\>"
color brightcyan "^[[:space:]]*#[[:space:]]*(define|include|(un|ifn?)def|endif|el(if|se)|if|warning|error)"
color brightmagenta "'([^'\]|(\\['"abfnrtv\\]))'" "\\(([0-3]?[0-7]{1,2}))'" "'\\x[0-9A-Fa-f]{1,2}'"

## GCC builtins
color green "__attribute__[[:space:]]*\(\([^)]*\)\)" "__(aligned|asm|builtin|hidden|inline|packed|restrict|
section|typeof|weak)__"

#Operator Color
color yellow "[.:;,+*|=!\%]" "<" ">" "/" "-" "&"

#Parenthetical Color
color magenta "[(){}]" "\[" "\]"

## String highlighting.  You will in general want your comments and
## strings to come last, because syntax highlighting rules will be
## applied in the order they are read in.
color cyan "<[^=        ]*>" ""(\\.|[^"])*""
##
## This string is VERY resource intensive!
#color cyan start=""(\\.|[^"])*\\[[:space:]]*$" end="^(\\.|[^"])*""

## Comment highlighting
color brightblue "//.*"
color brightblue start="/\*" end="\*/"

## Trailing whitespace
color ,green "[[:space:]]+$"
```

g6) Effectuer la même procédure pour le fichier ~/.nano/java.nanorc :

```
## Here is an example for Java.
##
syntax "Java" "\.java$"
color green "\<(boolean|byte|char|double|float|int|long|new|short|this|transient|void)\>"
color red "\<(break|case|catch|continue|default|do|else|finally|for|if|return|switch|throw|try|while)\>"
color cyan "\<(abstract|class|extends|final|implements|import|instanceof|interface|native|package|
private|protected|public|static|strictfp|super|synchronized|throws|volatile)\>"
color red ""[^"]*""
color yellow "\<(true|false|null)\>"
icolor yellow "\b(([1-9][0-9]+)|0+)\.[0-9]+\b" "\b[1-9][0-9]*\b" "\b0[0-7]*\b" "\b0x[1-9a-f][0-9a-f]*\b"
color blue "//.*"
color blue start="/\*" end="\*/"
color brightblue start="/\*\*" end="\*/"
color ,green "[[:space:]]+$"
```

6. Avec la commande de recherche, effectuer une commande de "recherche remplacement" du fichier texte pour rechercher les caractères "//" et les remplacer par "" :

CTRL + ALT GR + \
"//" puis entrer ""
Appuyer sur "o" pour "oui"

Que s'est-il passé ?

En java "//" représente les commentaires.
Ce sont des lignes qui ne sont jamais considérées par la machine lorsqu'elles sont compilées.