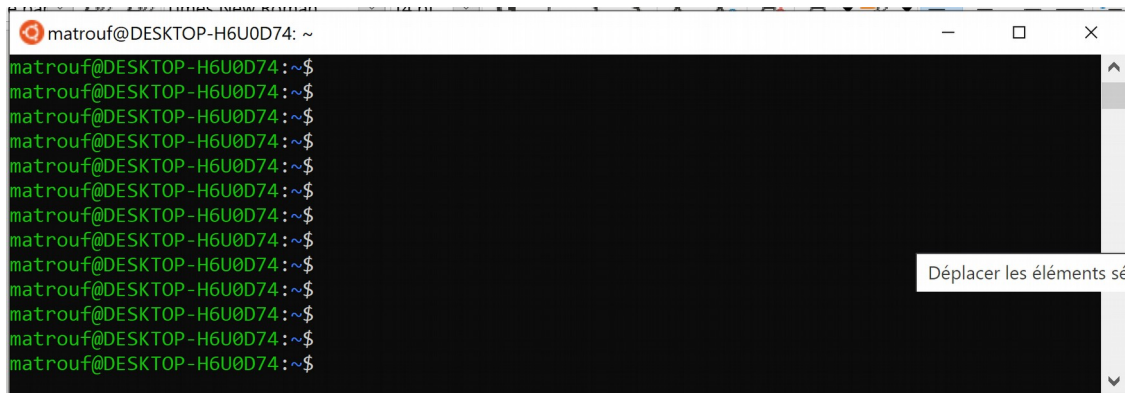


UAPV/CERI - BASES DE LA PROGRAMMATION L1 S1 INFORMATIQUE

Feuille de TP n° 1

Linux est le système d'exploitation utilisé au CERI (nous disposons de deux salles équipées avec windows). Le terminal permet de communiquer avec le système d'exploitation et de demander la réalisation de certaines tâches.

- Lancer un terminal : taper le mot « terminal » dans la barre de recherche et cliquer sur l'icône « terminal »



```
matrouf@DESKTOP-H6U0D74: ~  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$  
matrouf@DESKTOP-H6U0D74:~$
```

- **pwd** : permet de savoir où on se situe dans l'arborescence des fichiers
- **mkdir** : permet de créer un répertoire à partir du répertoire courant.
- créer un répertoire appelé BDP (Bases De Programmation) en utilisant la commande **mkdir BDP**
- **cd** : permet d'aller vers un répertoire : la commande **cd rep**, permet d'aller vers le répertoire **rep**
- aller dans le répertoire **BDP** en utilisant la commande **cd BDP**
- créer un répertoire appelé **TP1** : **mkdir TP1**

Pour la prochaine série TP2 vous créerez dans le répertoire **BDP** un répertoire appelé **TP2**



```
matrouf@DESKTOP-H6U0D74: ~/BDP/TP1  
matrouf@DESKTOP-H6U0D74:~$ pwd  
/home/matrouf  
matrouf@DESKTOP-H6U0D74:~$ mkdir BDP  
matrouf@DESKTOP-H6U0D74:~$ cd PBD  
-bash: cd: PBD: No such file or directory  
matrouf@DESKTOP-H6U0D74:~$ cd BDP  
matrouf@DESKTOP-H6U0D74:~/BDP$ pwd  
/home/matrouf/BDP  
matrouf@DESKTOP-H6U0D74:~/BDP$ mkdir TP1  
matrouf@DESKTOP-H6U0D74:~/BDP$ cd TP1  
matrouf@DESKTOP-H6U0D74:~/BDP/TP1$ pwd  
/home/matrouf/BDP/TP1  
matrouf@DESKTOP-H6U0D74:~/BDP/TP1$ geany exo1.cpp &
```

- la commande **geany exo1.cpp** va vous ouvrir un éditeur afin que vous puissiez rédiger vos programmes.

----- Saisissez dans l'éditeur les lignes suivantes :

```
// _____  
#include <iostream>  
  
using namespace std ;  
  
int main()  
{  
    cout<<"Hello World"<<endl ;  
    return(0) ;  
}  
// _____
```

----- Dans votre éditeur aller vers le menu fichiers et cliquer sur enregistrer (ou faites « **ctrl s** »)

----- Pour compiler votre programme `exo1.cpp`, tapez dans le terminal : **g++ `exo1.cpp` -o `exo1`**

----- cette commande crée à partir du fichier source `exo1.cpp` un exécutable appelé **`exo1`**

----- pour exécuter `exo1` on fait **`./exo1`** ce qui conduira à l'affichage : **`Hello World`**

Voici une liste de commandes très utiles

Les commandes de gestion des répertoires et des fichiers

`pwd` (affiche le chemin absolu du répertoire courant)
`ls` (list, affiche les répertoires et les fichiers du répertoire actif)
`ls` (affiche seulement les noms)
`ls toto*` (affiche les fichiers commençant par `toto`)
`ls -l` (affiche le format long : types + droits + Nbre de liens +)
`cd` (change directory)
`cp chemin` (vers le répertoire dont le chemin absolu est donné)
`cd ..` (répertoire parent)
`cd ~` (répertoire de base)
`cd -` (répertoire précédent)
`cd /` (répertoire racine)
`cp` (copie)
`cp rapport*.txt` sauvegarde
`cp * dossier` (copie)
`mv` (move, renomme et déplace un fichier)
`mv source destination`
`mv * dossier` (déplace tous les fichiers du répertoire actif vers le répertoire dossier)
`mkdir` (créer un répertoire)
`mkdir repertoire`
`rmdir` (effacer un répertoire)
`rmdir dossier` (supprime un répertoire vide)
`rm` (remove, efface!!!)
`rm -R` (enlèvement récursif!!!)
`rm fichier`
`rm -i fichier` (interactivement, avec demande de confirmation)
`rm -f fichier` (avec force, sans demande de confirmation)
`rm -r fichier` (avec récursivité, avec les sous répertoires)
`rm -rf dossier` (supprime le répertoire et tout son contenu, sans confirmation)

Les commandes de recherche

grep (recherche les occurrences de mots à l'intérieur de fichier)
grep motif fichier
grep -i motif fichier (sans tenir compte de la casse)
grep -c motif fichier (en comptant les occurrences)
grep -v motif fichier (inverse la recherche, en excluant le "motif")
grep expression /répertoire/fichier
grep [aFm]in /répertoire/fichier
grep "\\$" *.txt

Les commandes d'édition

more ("pager" qui affiche page par page sans retour en arrière, "h" affiche l'aide contextuelle)
more fichier
more fichier1 fichier2
more *.txt
cat (concaténation avec le code de fin de fichier eof=CTRL + D)
cat fichier-un fichier-deux > fichier-un-deux
cat -n fichier > fichier-numéroté (crée un fichier dont les lignes sont numérotés)
cat -nb fichier (affiche sur la sortie standard les lignes numéroté, sauf les lignes vides)
head (affiche les 10 premières lignes d'un fichier)
head -n22 fichier (affiche les 22 premières lignes)
vi (l'éditeur en mode texte universel)
emacs (l'éditeur GNU Emacs multi fonction pour l'édition, les mails, les news, la programmation, la gestion des fichiers,...)
xemacs (l'éditeur GNU Emacs sous X)
diff (différence entre deux fichiers, utiles pour chercher les modifications)
diff fichier1 fichier2

Tous les programmes doivent être écrits compilés et exécutés

EXERCICE 1

Écrire un programme qui demande à l'utilisateur 2 entiers qu'il met dans deux variable a et b et qui affiche « plus grand » si a est supérieur à b, sinon il affiche « plus petit ou égal ». Modifier le programme pour qu'il affiche : « plus grand », « plus petit », ou « égal ».

EXERCICE 2

Écrire un programme qui demande à l'utilisateur une longueur L et une largeur l (la longueur doit être supérieure ou égale à la largeur et $L \geq l$) et qui affiche la surface du rectangle correspondant. Sinon il affiche le message : « Données incohérentes »

EXERCICE 3

Écrire un programme qui lit un nombre entier et qui précise s'il est compris entre 10(exclu) et 20 (inclu) ou entre 50(inclu) et 100(inclu). Si le nombre vérifie la condition précédente alors il apparaît à l'écran le message suivant « dans l'intervalle », sinon il apparaît à l'écran le message suivant « en dehors de l'intervalle ».

EXERCICE 4

Écrire un programme qui lit 3 nombres **a**, **b** et **c** et les réordonne de façon à ce que **a** soit inférieur ou égal à **b** et **b** soit inférieur ou égal à **c**. Le programme affiche a, b et c pour vérification.

EXERCICE 5

Écrire un programme qui demande à l'utilisateur une opération (*,+,-,/), suivie de deux opérandes. Le programme doit afficher ensuite le résultat de l'opération appliquées aux deux opérandes. Les opérandes et les résultats doivent être des entiers.

Attention : pensez à interdire la division par 0.

Exécution :

Entrez l'opération : "*"

Entrez la première opérande : 3

Entrez la deuxième opérande : 4

Le résultat de 3*4 est 12

EXERCICE 6

Écrire un programme qui demande 3 valeurs **a**, **b** et **c** de type double et affiche les solutions de l'équation $ax^2+bx+c=0$. On vérifiera que a est différent de 0.

EXERCICE 7

Écrire un programme qui reçoit un nombre à 4 chiffres et qui affiche les chiffres un à un avec un espace les séparant. Le programme doit vérifier d'abord que le nombre contient 4 chiffres. Dans le cas contraire le programme affiche un message : « le nombre ne contient pas 4 chiffres »

EXERCICE 8

Écrire un programme qui demande à l'utilisateur de taper cinq entiers et qui affiche leur moyenne. Le programme ne devra utiliser que deux variables.

EXERCICE 9

Écrire un programme qui demande à l'utilisateur de taper 6 entiers et qui affiche la plus petite valeur et la plus grande valeur. Le programme ne devra utiliser que 3 variables.