UTC502 - TP n°2
Introduction à la programmation en C

1. Prise en main d'Unix

2.1. Définition

Unix est une famille de systèmes d'exploitation dont les principales caractéristiques sont: le multitâche et le multi-utilisateur. Initialement développé par Ken Thompson en 1969 dans les laboratoires Bell, Unix regroupe maintenant un large panel de systèmes : HP-UX, Solaris, FreeBSD, etc. Linux est un sous-ensemble des systèmes Unix dont la particularité est d'être libre : Debian, Red Hat, Slackware, SuSE, Ubuntu, etc.

2.2. Commandes de base

Le Shell est un interpréteur de commande permettant d'accéder aux principales fonctionnalités du système d'exploitation. Les principales commandes sont les suivantes :

- Manuel des commandes : `man.`
- Fichiers et répertoires : `cd, chmod, cp, ls, mkdir, mv, pwd, rm, rmdir, find.`
- Tâches :
  `bg, fg, kill, ps.`
- Utilisateur :
  `exit, passwd, who.`
- Variables d'environnement :
  `echo, printenv, set.`
- Edition : `cat, less, vi.`

2. Elément de programmation en C

3.1. Définition

Le C est un langage de programmation impératif qui est apparu en 1972 suite aux travaux de Dennis Ritchie et Ken Thompson. Le langage C a donné naissance par la suite au C++ grâce aux travaux de Bjarne Stroustrup.

3.2. Les phases de compilation

Les étapes permettant d'obtenir un exécutable à partir des fichiers sources sont, en pratique, les suivantes :

- *Pré-compilation* : Interprétation des directives du pré-processeur.
- *Compilation* : Analyse lexicale et syntaxique du code afin de produire un code proche de l'assembleur.
- *Assemblage* : Interprétation du code assembleur afin de produire un fichier en langage binaire (fichier objet d'extension « .o »).
- *Edition de lien* : Rassemblement des différents fichiers objets et bibliothèques statiques au sein d'un même fichier exécutable.

Nous nous servirons du programme gcc (GNU Compiler Collection) pour effectuer la compilation des programmes. Pour compiler un fichier source « source.c » et réaliser un exécutable binaire « executable » la ligne de commande est :

```
gcc source.c -o executable
```

Puis, l'exécution du programme est réalisé par la commande :

```
./executable
```

3.3 Eléments de syntaxe

Voici une liste des principaux éléments du langage C :

Commentaire :

`// Commentaire`	commentaire sur une ligne
`/* Commentaire */`	commentaire sur plusieurs lignes

Types de base :

`char`	caractère
`int`	entier
`long`	entier long
`float`	nombre à virgule flottante
`double`	nombre à virgule flottante
`struct`	structure

Opérateurs :

`=`	affectation
`+, -, *, /, %`	addition, soustraction, multiplication, division, modulo
`>, <, >=, <=, ==, !=`	supérieur, inférieur, supérieur ou égal, inférieur ou égal, égal, différent
`&&, ||`	et logique , ou logique
`&var`	adresse de
`*var`	indirection
`.`	sélection
`->`	indirection et selection
`sizeof`	taille de

Instructions :

`if (condition) {instructions}` `else {instructions}`	condition
`for (initialisation ; continuation ;` `iteration) {instructions}`	boucle explicite
`while (condition) {instructions}`	boucle implicite
`do {instructions}` `while (conditions)`	boucle implicite

`switch (expression) { case`
`expression_constante :`
`instructions [break] ...`
`default : instructions`
`}`	

`return resultat`	retour

Directives du pré-processeur :

`#include`	inclusion de fichier
`#define`	définition d'une macro
`#if, #else, #ifdef, #ifndef, #endif`	directives pour la compilation conditionnelle

Exercice n°1

Un programme de multiplication matrice-vecteur.

Question 1.

Ecrire un programme effectuant la multiplication d'une matrice par un vecteur.

- L'en-tête du fichier devra contenir le principe de l'algorithme comme il a été vu en cours.
- La matrice et le vecteur seront déclarées explicitement dans le code :

```
#define L 2
#define N 3
```

```
... float A[L][N] = {{1.0 , 2.5 , 3.0} , {2.0 , 3.0 ,
2.5}}; float B[N] = {0.5 , 1.0 , 0.0};
```

Question 2.

Ecrire une fonction permettant d'afficher un vecteur à l'écran et l'utiliser dans le programme précédent pour afficher le résultat.

- Le prototype de la fonction aura la forme suivante :
  `void affiche_vecteur(int nb_elements, float v[nb_elements]);`

Pour l'affichage, nous utiliserons la fonction « printf » de la librairie « stdio ». L'affichage d'un flottant se fait de la manière suivante : `float var = 2.5; printf("Valeur de la variable: %f \n", var)`

Question 3.

On réalise la question 1 et 2 pour la multiplication de matrices.

On rappelle ici l'exemple de multiplication matrice 3x3 par une autre matrice 3x3 :

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$
$$= \begin{bmatrix} a_1x_1+a_2x_2+a_3x_3 & a_1y_1+a_2y_2+a_3y_3 & a_1z_1+a_2z_2+a_3z_3 \\ b_1x_1+b_2x_2+b_3x_3 & b_1y_1+b_2y_2+b_3y_3 & b_1z_1+b_2z_2+b_3z_3 \\ c_1x_1+c_2x_2+c_3x_3 & c_1y_1+c_2y_2+c_3y_3 & c_1z_1+c_2z_2+c_3z_3 \end{bmatrix}$$

Question 4.

Permettre la saisie de l'utilisateur de la taille des matrices, avec un affichage d'erreur en cas de situations impossibles (incompatibilité lignes colonnes).

Exercice n°2

Utilisation de structures.

Question 1.

Ecrire un programme qui définit un type de structure nommée `produit` puis qui crée un tableau de produits contenant les données suivantes :

Produits				
code	3	5	7	6
quantite	0	1	2	10
prix	5.80	12.40	13.50	10.00

Question 2.

Ecrire une fonction qui permet l'affichage d'un produit.

- Le prototype de la fonction sera :
  `void affiche_produit(struct produit p);`

Question 3.

Compléter le programme de manière à ce qu'il affiche les produits dont la quantité est inférieure à un seuil donné.

- La lecture d'un nombre sur l'entrée standard s'effectuera par la fonction « scanf » de la librairie « stdio ». `int seuil; scanf("%d", &seuil);`

Question

Exercice n°3

Moyenne et écart type

Question 1.
Ecrire un algorithme permettant de calculer la valeur moyenne d'une séquence de valeur réelle. Les valeurs sont déclarées dans le main.

Question 2.
Modifier le programme pour intégrer l'écart type. On rappelle l'écart type comme étant :

$$\sigma = \sqrt{V} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

Question 3
Modifier le programme pour permettre la saisie de la séquence par un utilisateur (scanf) et de choisir l'écart type via un menu. Le programme permet de choisir entre l'écart type, la moyenne ou quitter.

Question 4
Rajouter une entrée dans le menu proposant de trier les valeurs rentrées par l'utilisateur de manière croissante ou décroissante.

Exercice N°4

Fonction random

La fonction rand() permet de générer des entiers aléatoires selon la loi uniforme entre [0, RAND_MAX].

Traiter les questions suivantes en permettant à l'utilisateur de fixer le nombre de valeurs générées en paramètre du binaire (en N=100, 1000 ou 10 000 par exemple).

Le programme prendra la forme d'un menu permettant de choisir ces différents calculs et éléments.

Question 1.
Observer les valeurs générer et les afficher sous une forme intelligible.

Question 2.
Donner le pourcentage de valeurs inférieurs à la moitié de l'intervalle (valeur médiane)

Question 3.
Même question pour le quart de la valeur et pour 75% de la valeur.

Question 4.
Généraliser l'approche en permettant de fixer le seuil de calcul par l'utilisateur.

Question 5.
Trouver le nombre de tirage nécessaire pour générer 5 valeurs de plus dans la première moitié que dans la deuxième.

Question 6.
A partir de la question précédente généraliser pour n valeurs.

Exercice 5

Gestion du temps

Ecrire le code d'un chronomètre indiquant le nombre de secondes écoulées depuis deux appui sur une touche du clavier.

On s'aidera :

- De la fonction kbhit(), qui renvoie un entier (code de la touche) lorsqu'une touche du clavier est activée
- La fonction clock(), qui renvoie un clock_t, comme timestamp au moment de l'appel de la fonction

Exercice 6

Interpréteur de commande

Question1.
Recoder un interpréteur de commande simple « comprenant » la fonction « more », permettant la lecture d'un fichier.

On utilisera fopen pour écrire le contenu d'un fichier sur la sortie standard.

L'exécution du programme donnera lieu à l'affichage d'un prompt qui interprétera la commande monmore avec un fichier donné en paramètre. Il donnera un message d'erreur pour tout autre commande tapée.

Question2.
Enrichir l'interpréteur de commande avec la commande monsleep qui prend un entier en paramètre. (boucler sur une condition de temps).

Question 3
Enrichir l'interpréteur de commande avec la commande monsort, qui trie des valeurs d'entier dans un fichier passé en paramètre et les affiche sur les sortie standard.

Exercice 7

Composition

Recoder le comportement d'un Dab (distributeur automatique bancaire) lors du retrait d'un client par rapport à une somme d'argent.

Le dab composera la somme demandée par le client avec les différentes coupures disponibles (5, 10, 20 et 50 euros) en donnant par ordre de priorité de grosses coupures.

Un menu permettra de demander à l'utilisateur la somme qu'il souhaite retirer, et indiquera le nombre de billets de chaque coupure.