# LO21 - TP n°1 Introduction à la programmation en C

## 1. Prise en main d'Unix

### 1.1. 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.

### 1.2. 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

### 2.1. 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.

### 2.2. 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
```

### 2.3 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}`<br>`else {instructions}` | condition |
| `for (initialisation ; continuation ;`<br>`iteration) {instructions}` | boucle explicite |
| `while (condition) {instructions}` | boucle implicite |
| `do {instructions}`<br>`while (conditions)` | boucle implicite |
| `switch (expression) {`<br>`case expression_constante :`<br>`instructions [break] ...`<br>`default :`<br>`instructions`<br>`}` | |
| `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);
```

# 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);
```