Résumé

Chapitre 1 : Introduction

Section 1.1 : Introduction

Un système unix est composé de trois grands types de logiciels:

1. **Le noyau du système d'exploitation** : chargé au démarage de la machine, il se charge de toutes les interactions entre les logiciels et le matériel.
2. **Les librairies** :Nombreuses, elles facilitent l'écriture et le développement d'applications.
3. **Les programmes utilitaires** : utilisés pour résoudre une série de problèmes.

Un processus : ensemble cohérent d'instructions qui utilisent une partie de la mémoire et sont exécutées sur un processeur.

mv : utilitaire pour renommer ou déplacer un fichier ou dossier.
head / **tail** : extrait le debut / la fin d'un fichier.
wc : compte le nombre de (lignes (-l), mots, caractères).
sort : trie le fichier par ordre alphabétique
uniq : retire les doublons *(Attention : fichier trié au préalable)*
cp : copie un fichier ou dossier (-r pour les dossiers).
pwd : affiche le répertoire courant.

Chapitre 2 : Langage C

Section 2.1 : Le langage C

préprocesseur: macros qui sont effectuées en amont de la compilation finale du programme.

Le manuel, accessible via la commande **man** se divise en :

1. Utilitaire disponible pour tous les utilisateurs
2. Appels systèmes en C

3. Fonctions de la librairie
4. Fichiers spéciaux
5. Formats de fichiers et conventions pour certains types de fichiers
6. Jeux
7. Utilitaires de manipulation de fichiers textes
8. Commandes et procédure de gestion du système

Section 2.2 : Types de données

- decimal : 123
- Binaire : **0b** 1111011
- Octal : **0** 173
- Hexadécimal :**0x** 7B

```
sizeof(DATA_TYPE) // taille en mémoire
```

Les **nombres signés** sont représentés sous la forme : Signe (négatif si = 1) - Nombre.
Les **nombres non-signés** sont représentés sous la forme binaire std $5 = 2^2 + 2^0$.
La **mémoire** est une zone qui est définie et accessible via son adresse. Un **pointeur** est une variable contenant l'adresse d'une autre variable.

```
    &var // adresse à laquelle une variable est stockée
    var // variable en mémoire
    *ptr // récupère la valeur à l'adresse du pointeur
```

En C, contrairement au java, **pas de garbage collection**. Il faut donc vider(**free**) la mémoire.

Une **structure** est une combinaison de différents types de données simples ou structurés.
Dans les premières version du langage, les structures avaient une taille fixe

- **Créer une structure**
  `struct NOM-STRUCTURE { int VARIABLE1; int VARIABLE2; };`
- **Créer une instance de la structure**
  `struct NOM-STRUCTURE NOM-INSTANCE = {1,2};`
- **Accéder à une variable en mémoire directement**
  `NOM-INSTANCE.VARIABLE1= 2;`
- **Accéder à l'élément d'une structure contenue dans un pointeur**
  `(* ptr).VARIABLE1 ptr->VARIABLE1`

Déclaration : Indique au compilateur le type des arguments et le type de la valeur de retour(en cas de fonctions). Toutes les fonctions et variables doivent être déclarées avant d'êtres utilisées.
Définition : Le corps de la fonction est spécifiée dans la déclaration ou dans le cas d'une variable déans son initialisation.

Section 2.3 : Declarations

La portée d'une variable : partie du programme ou la variable est accessible et où sa valaur peut-être modifiée.

Section 2.5 : L'organisation de la mémoire

La mémoire peut-être divisée en **six zones** principales :

- **Le segment text** : Contient toutes les **instruction** qui sont exécutées par le microprocesseur. (uniquement accessible en lecture)
- **Le segment des données initialisées** : Contient l'ensemble des données et chaînes de caractères qui sont utilisées dans le programme. (il comprend l'ensemble des variables globales déjà initialisées)
- **Le segment des données non-initialisées**: Contient les valeurs des variables non-globales ou les variables globales non initialisées. Elle est initialisée à 0 lors du démarage du programme. Attention toute fois aux variables locales qui ne sont pas explicitement intialisées
- **Le tas (ou heap)**: C'est dans une des 2 zones dans laquelle un programme peur obtenir de la mémoire supplémentaire pour y stocker de l'information. Un programmeur peut réserver une zone permettant de stocker des données et y associer un pointeur. (brk(2) et sbrk(2) modifient la taille du heap). On associe ainsi un pointeur à l'adresse réservée par le programme.

En pratique, on utilise :
`string= (char * ) malloc(length * sizeof(char)) free(string);`

- **La pile (ou stack)**: Cette zone est très importante, elle stocke : l'ensemble des variables locales, les variables de retour de toutes les fonction qui sont appelées, les arguments placés aux fonctions.