PROGRAMMATION AVANCÉE
TP3
MANIPULATIONS DE CHAÎNES ET FONCTIONS VARIADIQUES

1. Manipulations de Fichiers et Manipulation de chaînes

Exercice 1 Concevez un programma qui écrit dans un fichier passé en paramètre. Le programme écrira un chaîne pas en paramètre 2000 fois. Entre chaque écriture on pour *endormir* le programme avec la fonction `usleep( int millisecondes)` définie dans `unistd.h`[1]. On pourra choisir un temps d'endormissement aléatoire en utilisant `rand`.

Exécutez votre programme sur un fichier particulier et dans un même temps exécutez le même programme sur le même fichier avec une autre chaîne.

Pour lancer quasiment en même temps les deux programmes on pourra exécutez les commandes suivantes :

```
prg fichier chaine1 & prg fichier chaine2 &
```

Quel comportement observez vous ?

Exercice 2 Une chaîne de caractère contient des entiers, écrivez une fonction découpage qui crée un tableau de chaînes de caractères qui contient chaque entier. On pourra utiliser la fonction `isdigit()` définie dans `ctype.h`

Par exemple pour la chaîne "`23 67 89 -56`" On aura le tableau qui contiendra les valeurs 23 67 89 et -56.

Il sera donc nécessaire de disposer d'une fonction qui compte le nombre de nombres représentés.

Votre fonction devra renvoyer `NULL` si la chaîne contient quelque chose qui n'est pas reconnu comme un entier.

Exercice 3 En utilisant les fonction définies dans `unistd.h` du même type que `isdigit`. Écrivez un programme qui compte le nombre de lignes présents dans un fichier, le nombre de caractères, et le nombre de mots. En somme, votre programme fournira le même type d'informations que la commande `wc` (pour word count).

2. Fonctions Variadiques et Macros

Exercice 4 Écrivez une fonction qui fait la moyenne de plusieurs valeurs entières passés en paramètres.

Que se passe-t'il si un des paramètres passé à la fonction est de type flottant par exemple ? Est-ce que le programme compile ? Qu'en est-il de l'exécution ?

Que se passe-t'il si la fonction reçoit moins d'arguments que ce qui est attendu ?

Exercice 5 Écrivez une macro `MAX` qui calcule le maximum de deux valeurs. Essayez votre fonction. Que se passe-t'il si l'on essaye de comparer des valeurs de types différents ?

Peut-on définir une macro MAX qui prend trois éléments ?
Pour la macro à trois éléments peut-on utiliser la macro à 2 éléments et faire la composition des deux (*p. ex.* `MAX(A,MAX(B,C))`). En arrêtant la compilation après le passage du pré-processeur (`gcc -E fichier.c`) expliquez ce qu'il se passe.

1. unistd pour Unix standard donc non portable

3. Bonus

Exercice 6 On dispose d'une chaîne de caractère qui représente une expression arithmétique en notation Polonaise inversée. Le but est d'écrire une ensemble de fonction qui vérifiera que l'expression est syntaxiquement correcte, et procédera à son évaluation.

Exemple, L'expression (45*6)+3 s'écriera `(+ (* 45 6) 3))`

Plus précisément, une expression commencera toujours par `(` et finira par `)` Le premier symbole qui suivra `(` sera un opérateur (`+ - * / %`...) Et ensuit des nombres où des expressions. Par exemple `(+ 1 2 3 (* 4 5 6) (- 12))` consiste en l'opération suivante : $1 + 2 + 3 + (4 * 5 * 6) + (-12)$.

L'avantage de cette notation est que la priorité des opérateurs est clairement définie et permet de limiter le nombre de symbole.

(1) Écrivez une fonction qui détermine si l'expression est correcte (p. ex `(* 4)` n'est pas correcte car la multiplication attend au moins 2 paramètres).

(2) Écrivez une fonction qui, si l'expression est correcte, évalue l'expression et renvoie la valeur (p. ex pour `(+ 1 2 3)` votre fonction devra renvoyer 6).

(3) Adaptez vos fonction pour que pouvoir traiter la notation Polonaise (Dans la notation polonaise, l'opérateur est à la fin ; exemple `(1 2 3 +)`)

(4) Que l'on puisse définir ses propres fonctions : par exemple `sqrt` qui renverra la racine carrée d'un nombre.