TP 3 – Programmation avancée en C version 2

Exercice 1. ***Pointeurs et paramètres du programme***

Lorsqu'un programme est exécuté en ligne de commande, les paramètres du programme sont passés à la fonction `main` *au moyen d'un tableau de chaînes de caractères* `argv`*. Le nombre d'éléments dans ce tableau est donné par un entier* `argc`*. Quand on écrit un programme C, on peut donc écrire :*

`int main(void)`*, si l'on ne souhaite pas récupérer les paramètres du programme,*

`int main(int argc, char ** argv)`*, si l'on souhaite les récupérer.*

1. Ecrivez un programme nommé `parametres` qui affiche le nombre d'arguments passés. Testez le par exemple en appelant `parametres`, `parametres abc`, `parametres abc 123`, etc.

2. Ecrivez un programme qui affiche tous ses paramètres passés à la fonction `main` (un paramètre par ligne).

3. Ecrivez un programme qui prend en paramètres de la fonction `main` un nom, un prénom, un âge et les mémorise dans une structure (`struct`). Si l'age est incorrect (n'est pas un entier compris entre 0 et 127), le programme essaiera avec le(s) éventuel(s) paramètre(s) suivant(s) de la ligne de commande jusqu'à en trouver un correct, ou mettra 0 s'il arrive à la fin des paramètres sans succès.

Exercice 2. ***Fichiers***

Les fichiers se manipulent au moyen d'un type appelé `FILE` *(il s'agit en fait d'une structure pour laquelle le mot-clef* `struct` *est omis), et des fonctions telles que* `fopen`*,* `fclose`*,* `fprintf`*,* `fscanf`*,* `fputc`*,* `fgetc`*.*

1. Consulter la documentation de ces quatre fonctions (en tapant `man 3 fopen` par exemple, ou en cherchant sur Internet) afin de savoir les utiliser.

2. Le programme `cat` affiche un fichier texte à l'écran. Ecrire un programme semblable à `cat`. Il est bien sûr conseillé de ne pas appeler son programme `cat`!
N.B. le programme se comporte-t-il différemment selon qu'on utilise la fonction fscanf ou la fonction fgetc pour lire un fichier texte contenant des espaces et des retours à la ligne ?

3. La commande `cp` copie un fichier `source` dans un fichier `destination`. Ecrire un programme semblable à `cp`, les noms de fichiers source et destination étant passés en argument. Il est bien sûr conseillé de ne pas appeler son programme `cp`. On pourra réinvestir une partie du code créé précédemment : quelle(s) fonction(s) changent ?

Exercice 3. ***Erreur de pointeur***

1. Ecrire un programme qui initialise un pointeur `p` à `NULL`, et qui tente d'accéder à la variable pointée par ce pointeur. Que se passe-t-il à l'exécution ?

2. Compiler en ajoutant l'option `-g` de `gcc`, puis chercher l'erreur avec `gdb` étudié précédemment.

3. Afin d'éviter cette erreur, Initialiser le pointeur `p` avec la fonction malloc. A-t-on le choix de la taille d'allocation ?

4. Ajouter une variable `int i` dans le programme et lui faire afficher la valeur du pointeur pointant sur cette variable `i`. Faire afficher aussi la valeur du pointeur `p` avant puis après allocation.

Exercice 4. ***Pointeurs et files***

1. Implémenter une structure de file d'entiers positifs en utilisant une liste (et des pointeurs)

2. Implémenter les opérations suivantes :

a. `creer` : Créer une file vide ;
b. `estvide` (fonction booléenne) : Déterminer si la file est vide ;
c. `enfiler` : Ajouter un élément en fin de file ;
d. `defiler` : Retirer (et retourner) l'élément en tête de file (que retourner si la file est vide ?) ;
d. `refiler` : Donner (sans le retirer de la file) la valeur de l'élément en tête de file ;
e. `taille` : Indiquer le nombre d'éléments de la file.

3. Faire un programme qui utilise cette file de la manière suivante :

1^o) Le programme demande à l'utilisateur de saisir une suite d'entiers positifs et les place à mesure dans la file. N.B. Comment indiquer au programme que la saisie est terminée ?
2^o) Le programme affiche dans l'ordre normal de sortie la suite d'entiers de la liste (l'affichage du programme ne doit bien sûr commencer qu'une fois que l'utilisateur a saisit tous ses nombres). A la fin de l'affichage, la liste sera vide.

4. Modifier ce programme afin que la liste de nombres puisse être saisie comme paramètres de ligne de commande. S'il n'y a aucun paramètre, alors le programme demandera à l'utilisateur la liste de nombre comme précédemment.

5. Modifier le programme pour qu'il conserve en mémoire, dans le bon ordre, les éléments de la liste après leur affichage. N.B. Créer une deuxième liste par exemple.