Université Grenoble Alpes **Année 2018/2019**
L3 MIAGE

Introduction au Système
Planche de TP n°3

Exercice 1. (rappels C)

Définir un type **structure `personne`** permettant de représenter une personne par son nom (chaîne de caractères) et son age (entier).

On souhaite écrire le programme C qui saisit au clavier une quantité quelconque B d'informations sur des personnes, enregistre ces personnes dans un **tableau** (dynamiquement dimensionné pour ne contenir que B éléments), et affiche le nom de la plus agée de ces B personnes. Le nombre B de personnes à traiter sera **passé en paramètre** à la commande. Ecrire au préalable les fonctions suivantes (déterminer vous-même l'entête de ces fonctions) :

- fonction `saisie_personne_suivante` : saisit au clavier **un** nom et **un** entier (attendu positif), et renvoie en résultat **la** structure `personne` correspondante. Elle devra demander à l'utilisateur de recommencer tant que l'entier saisi ne sera pas positif
- fonction `calcule_max` : saisit au clavier successivement les informations pour B personnes, stocke ces personnes dans le tableau, et renvoie en résultat l'indice dans le tableau de la personne la plus agée
- fonction `main` qui s'assure que la commande a reçu exactement 1 argument (sinon on affichera un message d'erreur et on s'arrêtera), interprète cet argument comme le nombre B de personnes à saisir[1], puis affiche le nom de la personne la plus agée.

On aura par exemple à l'exécution :

```
$ ./calcul_de_max 3
Saisir le nom :  Berthe
Saisir son age (entier positif) :  25
Saisir le nom :  Marcel
Saisir son age (entier positif) :  33
Saisir le nom :  Gertrude
Saisir son age (entier positif) :  -17
Attention, saisir un entier positif :  17
La personne la plus agee est : Marcel
```

Exercice 2. (rappels C - Makefile)

1. S'assurer que vous avez correctement réalisé l'exercice 1 de la planche 2.

2. Reprendre cet exercice en créant correctement les **fichiers** suivants :

- un fichier `exo2_fcts.c` contenant la fonction `remplir` et la fonction `calcule_min`, et son fichier de déclaration `exo2_fcts.h`
- un fichier `exo2.c` contenant la fonction `main` du programme

Penser à placer les bonnes directives aux bons endroits.

3. Taper les **commandes** permettant de produire les fichiers objets (.o) qui seront nécessaires à la création de l'exécutable, puis la commande permettant de créer cet exécutable.
Comment devez-vous vous y prendre pour recompiler (efficacement) le programme si seul le fichier source `exo2.c` a été modifié ?

4. Créer un **`Makefile`** pour la compilation de cette application.

Exercice 3. (rappels C - Makefile)

1. (Cf exercice 14 de la mise à niveau) Ecrire les **fonctions** de conversion suivantes :

- une fonction *convert_vers_bin* qui renvoie en résultat un vecteur de caractères représentant le codage binaire d'un nombre entier positif x, sur un nombre donné n de

[1] On pourra regarder le man de la fonction atoi

bits :

```
char *convert_vers_bin(int x, int n);
```

- une fonction *convert_vers_dec* qui renvoie en résultat un nombre entier positif correspondant à un vecteur de bits (chaque bit étant représenté par le caractère '0' ou '1') d'une taille donnée :

```
int convert_vers_dec(char v[], int n);
```

On prendra comme convention que les bits sont représentés du poids faible vers le poids fort. On ne devra pas utiliser de fonction puissance.

On définira de plus une fonction pour **l'affichage** d'un vecteur de bits (sous forme de caractères), afin d'afficher le résultat de *convert_vers_bin.*

Votre fichier contiendra au préalable la **définition** (par #define) d'une valeur N pour le nombre de bits souhaité pour la représentation des vecteurs (fixée par exemple à 8) qui servira de paramètre effectif pour le paramètre formel n.

Ecrire le programme qui saisit au clavier un nombre entier positif et affiche à l'écran sa valeur en binaire, puis qui saisit au clavier une suite de caractères '0' et '1' représentant une valeur binaire donnée à partir du bit de poids faible et qui calcule et affiche la valeur décimale correspondante. On aura par exemple à l'exécution :

```
-- Essai de conversion decimal -> binaire --
Saisir la valeur a convertir en binaire :  83
Resultat converti :
0 1 0 1 0 0 1 1
-- Essai de conversion binaire -> decimal --
Saisir la chaine a convertir en decimal (poids faible -> fort) : 11001010
Valeur convertie = 83
```

2. Nous allons maintenant organiser **modulairement** notre application, et réaliser le **Makefile** associé. Dans le répertoire courant, créer un répertoire *include* et y placer les fichiers fct_aff.h (contenant la déclaration de la fonction d'affichage) et fct_convert.h (contenant les déclarations des deux fonctions de conversion décimal <-> binaire).
Dans le répertoire courant, créer les fichiers correspondants fct_aff.c et fct_convert.c, ainsi qu'un fichier main.c ne contenant que la fonction main. Penser à placer les bonnes directives d'inclusion aux bons endroits, et à définir N au bon endroit.
Ecrire le Makefile associé :

- il définit les variables CC (le compilateur utilisé), SRCS (les fichiers sources), OBJS (les fichiers objets) définie par substitution de suffixe à partir de SRCS, et CFLAGS (ici option de compilation permettant de trouver les fichiers .h) qui sera utilisée par la règle implicite .c.o
- il contient une règle permettant de générer l'exécutable de l'application, à partir des fichiers objets. Il pourra aussi contenir une règle de "nettoyage" (cible clean).

Vous observerez attentivement, à l'exécution de votre Makefile, les commandes exécutées, de façon à bien comprendre ce qui se passe. Vous pourrez également tester en ayant modifié au préalable par exemple un seul fichier source : observer la différence par rapport à l'exécution lorsque tous les fichiers sources sont "neufs" (ou lorsqu'aucun fichier objet n'existe encore).

3. Supprimer maintenant la définition de N par le #define. Que se passe-t-il ? Pourquoi ?
Ajouter ce qui sera nécessaire dans votre Makefile pour que la macro N soit définie à la compilation[2].

[2] On pourra trouver des compléments d'information par exemple ici :
http://gcc.gnu.org/onlinedocs/gcc/Preprocessor-Options.html
http://www.rapidtables.com/code/linux/gcc/gcc-d.htm