Cours/TP6

1. Travail à faire :

1) Récupérer le fichier « ctp6_1.h », « ctp6_1.c » et « ctp6_1main.c ».

Construisez un projet avec ces trois fichiers si vous êtes sous CodeBlocks et essayez d'exécutez le fichier contenant le « main » (« ctp6_1main.c »).

Corrigez les 3 erreurs du « main » en mettant en commentaires les trois instructions concernées et exécutez le « main ».

Pourquoi dans le main ne pouvez-vous pas accéder à « mon_compte_secret » ?

Pourquoi dans le main ne pouvez-vous pas accéder à la fonction « je consommelocal » ?

Pourquoi la fonction « jeteste » y a accès ?

2) Récupérer le fichier « ctp6_2.h », « ctp6_2.c » et « ctp6_2main.c ».

Complétez le fichier « ctp6_2.c » .

Le « main » utilise des paramètres passés lors de l'exécution (voir partie du cours « passage de paramètres au main »). Ici, le paramètre sera le nom physique du fichier (nom complet avec l'extension).

Pour compilez et exécutez ce programme, vous n'utiliserez pas CodeBlocks. Dans un terminal, placez-vous dans le répertoire contenant les trois fichiers précédents.

Lisez le fichier « TP makefile.pdf » jusqu'à la partie « 2.2 » (vous pourrez appliquer la partie « 2.3 » quand vous aurez fini tout le tp pour vous exercez à la création et utilisation d'un makefile). Appliquez la partie « 2.1 » ou « 2.2 » pour compiler votre programme. Exécutez le programme exécutable obtenu en passant le nom d'un fichier texte non existant.

3) Le langage C fournit un en-tête spécialisé pour la gestion des erreurs : <errno.h>.

Cet en-tête déclare notamment une variable globale errno, et un certain nombre de codes d'erreur, qui permettent aux fonctions de la bibliothèque standard de reporter précisément la cause d'une erreur.

Il est nécessaire de placer errno à 0 *avant* d'utiliser la fonction qui peut échouer, car les fonctions de la bibliothèque standard ne sont pas obligées de la mettre à zéro en cas de succès.

Pour connaître la signification textuelle d'une valeur de errno , on utilise la fonction :

```
char * strerror(int code); de <string.h>
```

Une autre fonction, perror, permet d'envoyer le message correspondant à errno sur le flux des erreurs stderr.

```
#include <errno.h>
void perror(const char *chaine);
```

Récupérer le fichier « ctp6_3.c », le compiler et l'exécuter.
Appliquez l'utilisation de errno à l'exercice précédent.