Corrigé TP2

1 installeTP.sh

Normalement, à l'issue du premier TP, vous devez avoir dans votre répertoire personnel (~INF203/TP1/scripts) le script `installeTP.sh`. Vous pouvez le copier directement dans votre dossier personnel (~installeTP.sh) car il vous servira désormais à chaque début de TP à récupérer les fichiers dont vous aurez besoin. Ainsi, au début du TP2, vous pouvez taper la commande suivante :

```
./installeTP.sh 2
```

Où le 2 désigne le TP2. Cela remplace avantageusement la commande que vous aviez dû taper la semaine dernière pour récupérer les fichiers du TP1, à savoir :

```
cp -R /Public/203_INF_Public/TP1 INF203          # Ne plus jamais faire ça !
```

2 Compilation

Pour commencer le TP2, vous devez être au clair avec la **compilation**. La compilation, c'est l'étape où votre programme qui se trouve dans un fichier `.c` (comme par exemple `deborde_char.c`) va être traduit en instructions compréhensibles par l'ordinateur.

Avant :

```c
#include <stdio.h>

int main() {
   printf ("Hello, world!\n");
   return 0;
}
```

Après :

```
7f 45 4c 46 02 01 01 00  00 00 00 00 00 00 00 00
03 00 3e 00 01 00 00 00  80 05 00 00 00 00 00 00
40 00 00 00 00 00 00 00  38 22 00 00 00 00 00 00
00 00 00 00 40 00 38 00  09 00 40 00 24 00 23 00
06 00 00 00 05 00 00 00  40 00 00 00 00 00 00 00
```

Pour cela, on utilise la commande `clang` (Pour les anciens, il s'agit d'un équivalent à la commande `gcc`). Pour compiler le fichier `monprog.c`, par exemple, on exécutera donc la commande :

```
clang monprog.c
```

Cela créera un exécutable nommé `a.out`. On peut aussi entrer la commande suivante :

```
clang monprog.c -o mon_executable
```

Cela compilera de la même manière, mais en nommant l'exécutable produit : `mon_executable`. On pourra ensuite lancer notre programme en entrant la commande :

```
./mon_executable
```

On peut maintenant commener à répondre aux questions! Ouf!

3 Testons des entiers

[a] Pour ne pas retaper à chaque fois les mêmes commandes, on peut utiliser les flèches du haut et du bas pour remonter dans l'historique des commandes.

On peut aussi utiliser la touche TAB pour compéter dès que possible les noms de fichiers ainsi que les commandes que l'on est en train de taper.

[b] On pars du programme `exemple_generation.c` que l'on a un peu modifié pour qu'il affiche "Trop grand", "Trop petit" ou "Youpie!" :

```c
#include <stdio.h>
#include "generer_entier.c"

int main() {
    int x = generer_entier(100);
    printf("Entier généré : %d\n", x);

    if (x < 42) {
       printf("Trop petit !\n");
    } else if (x > 42) {
       printf("Trop grand !\n");
    } else {
       /* Si x n'est ni < 42 ni > 42, c'est qu'il est égal à 42 ! */
       printf("Youpie !\n");
    }

    return 0;
}
```

Une fois compilé, si on lance le programme plusieurs fois, on voit que l'on tombe souvent sur "Trop grand" et "Trop petit", mais rarement sur "Youpie!". On souhaite donc que le programme fasse lui même un certain nombre de générations. On utilise pour cela une boucle.

```c
#include <stdio.h>
#include "generer_entier.c"

int main() {
   int nombre_boucle = 20;

   for (int i = 0; i < nombre_boucle; i++) {
      int x = generer_entier(100);

      if (x < 42) {
         printf("Trop petit !\n");
      } else if (x > 42) {
         printf("Trop grand !\n");
      } else {
         /* Si x n'est ni < 42 ni > 42, c'est qu'il est égal à 42 ! */
         printf("Youpie !\n");
      }
   }

    return 0;
}
```

NB : La boucle `for` part de `i = 0` et s'arrête quand `i` n'est plus `< nombre_boucle`.
Donc `i` va de `0` à `nombre_boucle - 1`.

4 Provoquons un débordement

[c] [d] [e] Pour les trois précédentes questions, voilà le tableau résumé :

Type	Taille (octet)	Taille (bit)	Valeur Max	Temps débordement
unsigned char	1	8	0 .. 255	< 0.005 s
unsigned short	2	16	0 .. 65335	< 0.005 s
unsigned int	4	32	0 .. 4294967295	env. 10 s

[f] Comme l'ordinateur compte à la même vitesse quelque soit le type de la variable, on s'attend à ce que `deborde_int` prenne $\frac{\texttt{taille(int)}}{\texttt{taille(short)}}$ fois plus de temps que `deborde_short`, soit 65536 fois plus de temps. Cela dit, un temps d'exécution inférieur à 0.005 secondes n'est pas significatif. On ne peut donc pas vraiment déduire quoi que ce soit des mesures précédentes ...

5 Rangeons !

[g] Appel effectif à la fonction `inserer` :

```
inserer(T, i, valeur);
```

[h] Listing complet de `exemple_generation_tableau.c` :

```c
#include <stdio.h>
#include "generer_entier.c"

// affiche a l'écran T[0..nb-1]
void afficher(int T[], int nb) {
   int i;
   printf("[ ");
   for (i = 0; i < nb; i++) {
      printf("%d ", T[i]);
   }
   printf("]\n");
}

void echanger(int Tab[], int i, int j) {
   int tmp;
   tmp = Tab[i];
   Tab[i] = Tab[j];
   Tab[j] = tmp;
}

/* inserer a sa place  l entier  val dans la sequence triee Tab[0..nb-1] */
void inserer(int Tab[], int nb, int val) {
   Tab[nb] = val;
   int i = nb;
   while (i > 0 && Tab[i-1] > Tab[i]) {
      echanger(Tab, i - 1, i);
      i--;
   }
}

int main() {
   int Taille = 20;
   int T[Taille];
   int nb = 0, i = 0, valeur = 0;

   for (i = 0; i < Taille; i++) {
      valeur = generer_entier(100);
      inserer(T, i, valeur);
      nb++;
      afficher(T, i);
   }

   return 0;
}
```

6 Une (petite) commande de la semaine : `wc`

[i] La commande `wc` pour *word count*, permet de compter les lignes d'un fichier avec l'option `-l`. Exemple :

```
wc -l fichier.c
```

On note que `wc` ne fait que compter le nombre d'occurence du caractère de retour à la ligne "`\n`". Donc si la dernière ligne n'est pas suivie d'un retour à la ligne, elle n'est pas comptée.

[j] Pour compter toutes les lignes de code C que l'on a produites, on peut utiliser la commande suivante :

```
wc -l *.c
```

7 Installons le TP

Revenons sur les droits d'accès. On sait qu'il peut y avoir plusieurs utilisateurs sur un système UNIX comme Turing. En l'occurence, chaque étudiant a un compte. Il peut y avoir également plusieurs groupes. Chaque utilisateur peut appartenir ou non à chacun des groupes. On peut lister les groupes auxquels on appartient en tapant la commande :

```
$ groups
corentin cdrom floppy sudo audio dip video plugdev netdev
bluetooth lpadmin scanner
```

Dans cet exemple, on voit que sur mon ordinateur personnel, j'appartiens entre autres au groupe `cdrom`, ce qui me permet d'accéder au lecteur de CD, mais aussi au groupe `audio`, ce qui me permet d'écouter de la musique et d'enregistrer des sons.
Sous UNIX, chaque fichier appartient à UN utilisateur et à UN groupe. Par exemple pour le fichier suivant :

```
$ ls -l logo.jpg
-rw-r--r-- 1 corentin users 275085 avril 19  2017 logo.jpg
```

On voit que le fichier appartient à l'utilisateur `corentin` et au groupe `users`. On voit également que les droits sont définis de la manière suivante :

```
rw-r--r--
```

Soit :

u			g			o		
r	w	x	r	w	x	r	w	x
X	X		X			X		

Donc, l'utilisateur `corentin` peut lire et modifier le fichier et les membres du groupe `users` et le reste des utilisateurs du système (donc tous ceux qui ne sont pas dans le groupe `users`) ne peuvent que le lire.

[k] Dans cet exemple, l'utilisateur `NOM_A` essaye de copier le fichier `installeTP.sh` qui se trouve dans son répertoire personnel vers le répertoire personnel de `NOM_B`. Ce n'est pas possible car seul `NOM_B` a les droits nécessaires pour écrire dans son répertoire personnel. Les autres utilisateurs peuvent voir ses fichiers, les lire, mais pas les modifier. Vous pouvez vérifier cela en tapant la commande `ls -l` dans votre répertoire personnel. Vous verrez que tous vos fichiers et répertoires ont pour propriétaire vous-même et que leur droits interdisent la modification pour tous les autres utilisateurs (pas de droit `w`).

[l] Pour récupérer le fichier `installeTP.sh` sur le compte de votre binôme, on est donc obligé de procéder à l'inverse. On se connecte en tant que `NOM_B` et on copie `installeTP.sh` depuis le répertoire de `NOM_A`. On a le droit d'effectuer cette copie car on dispose des droits en lecture sur les fichiers de `NOM_A` (droit `r`) et que la copie ne modifie pas le fichier original. On entre donc la commande :

```
cp /home/n/NOM_A/installeTP.sh ~
```