TP4

La page web du cours se trouve ici :

`http://www.lsv.ens-cachan.fr/~schwoon/enseignement/systemes/ws1516/`.

1 Opérations sur les bits

Le langage C possède des opérateurs pour manipuler la représentation binaire des types entiers. Soient x, y deux variables parmi les types `char`, `short int`, etc, où x_i, y_i sont les i-ème bits (0 étant le bit le moins significatif). Soit $\otimes$ un opérateur, alors le résultat de $x \otimes y$ est le mot z tel que $z_i = x_i \otimes y_i$.

Les opérateurs binaires disponibles en C sont `&` (et), `|` (ou), `^` (ou exclusive), `~` (non). Attention, ceux-ci ne doivent pas être confondu avec les opérateurs *logiques* `&&`, `||`, etc qui testent simplement si les arguments sont zéro ou non. Du coup, `4&2` égale 0 mais `4&&2` égale 1.

Il existe des raccourcis tels que `x|=2` pour `x=x|2` ou `x^=y` pour `x=x^y`. De plus, il existe des opérateurs pour le *décalage* (`<<`, `>>`), p.ex.. `4<<2` égale 16.

1. Que font les instructions suivants si x, y sont des types entiers ?
   ```
   x ^= y;  y ^= x;  x ^= y;
   ```
2. Supposons que c, n sont des entiers. Quelle sera la valeur de c en fonction de n après le fragment suivant : What value does `c` take as a function of `n`?
   ```
   for (c = 0; n != 0; n &= (n-1)) c++;
   ```
3. Suivant (b), quelle condition sur n est vérifiée si l'expression suivante est non-zéro ?
   ```
   n & (n-1)
   ```

2 Virgule flottante

Rappel : en C, le type `float` représente des valeurs réelles selon le standard IEEE 754 dans la variante de 32 bit, avec 1 bit pour le signe, 8 pour l'exposant et 23 pour la mantisse. On considère le type suivant qui représente ces composants par trois entiers :

```
typedef struct { int signe; int exposant; int mantisse; } fc;
```

1. Écrivez une fonction C qui décompose un `float` dans ses trois composants. (Autrement dit, le paramètre d'une telle fonction est un `float`, et elle renvoit un `fc`. Par exemple, la rfrepresentation en IEEE 754 de 2.5 est de
   ```
   0 . 1000 0000 . 010 0000 0000 0000 0000 0000
   ```
   Dans ce cas, la structure renvoyée contiendrait `sign` $= 0$, `exposant` $= 0x80 = 128$ et `mantisse` $= 0x400000 = 4194304$.
   Rappel : Pour ce faire, il convient de se servir de la conversion des types en C (*typecast*), p.ex., `(int)f`, où `f` est un `float`, va interpréter le contenu binaire de `f` comme un entier. Rassurez-vous d'abord que `int` en a la même taille sur votre machine !

2. Créez une fonction qui fait l'inverse, c'est à dire qui renvoie le `float` correspondant à une structure `fc` donnée.
3. Réalisez l'addition réelle sur la base de l'addition des entiers, en passant par les structures `fc`. Pour simplifier, on fera les réstrictions suivantes : (i) les deux opérandes sont positifs ; (ii) on ne traite pas les débordements, (iii) ni les cas spéciaux NaN/Inf etc.

   L'addition dans le type `fc` se fait en trois étapes :

   (a) Uniformiser les deux valeurs, c'est à dire si les deux exposants sont différents, on ajuste la mantisse d'une des deux selon la différence.

   (b) Faire la somme des deux mantisses, en tenant compte du bit "caché" représentant la 1.

   (c) Normalizer la mantisse pour quelle soit dans $[1, 2)$, tout en ajustant l'exposant du résultat.

3 Linking

On considère le fragment suivant d'un programme de C :

```
x=y+z;
printf("x=%d\n",x);
```

En analysant le code assembleur produit par le compilateur on trouve que la première instruction va directement être réalisée par des opérations sur les registres et la mémoire tandis que la deuxième fait appel à une fonction externe.

`printf` est en effet rèalise par une fonction fournie par une *bibliothèque partagée*. Le modle de compilation préféré de Linux/Unix et d'autres systèmes est de stocker les programmes exécutables dans les fichiers sous une forme incomplète, tout en spécifiant quelles fonctions partagées seront nécéssaires à son exécution. Lors de l'exécution, le programme sera chargé dans la mémoire, puis un *éditeur des liens* établit les connexions entre le programme et les bibliothèques partagée (*linking* en anglais).

Normalement, le système sait trouver où trouver les bonnes librairies. Le comportement de l'éditeur des liens peut être modifié par deux variables :

— La variable `LD_LIBRARY_PATH` (qui peut être modifié dans la ligne de commande) donne une liste de répertoires où l'éditeur des liens pourra trouver des bibliothèques.

— La variable `LD_PRELOAD` définit quelques librairies dont l'utilisation sera prioritaire.

Dans cette exercises on va surtout explorer des application de cette derniÃ¨re que l'on va discuter lors d'une petite démonstration. Après, essayez le suivant avec les fichiers disponibles sur la page web :

1. On considère le programme `password1` (disponible sous forme binaire) qui demande un mot de passe. Trouvez le bon mot de passe.
2. Faisez le même pour `password2`.
3. Comment éviter les failles de sécurité présentes dans ces deux programmes ?
4. Le programme `game` vous demande une numéro entre 1 et 100 choisi aléatoirement lors de chaque exécution. Comment tricher pour gagner toujours au premier tour ?