TP11 : Pointeurs de fonction, utilisation de `gdb`

Programmation en C (LC4)

Semaine du 7 avril 2008

1 Les pointeurs de fonction

Un pointeur est une variable contenant une adresse mémoire. Les pointeurs peuvent également contenir l'adresse d'une fonction, c'est ce qu'on appelle un pointeur de fonction. Cette dernière peut ainsi être passée en paramètre à une autre fonction et être appelée.

Exercice 1 Utilisation simple des pointeurs de fonction

- Ecrire une fonction `fois_deux` qui multiplie par deux un entier
- Ecrire une fonction `appliquer_tableau(int f(int), int * t, int n)` qui applique une fonction `f`, au tableau `t` de taille `n`
- Ecrire une fonction `main` qui teste `appliquer_tableau`

▶ **Correction**

```
#include <stdio.h>
#include <stdlib.h>

int
fois_deux(int i)
{
  return 2 * i;
}

void
appliquer_tableau(int f(int), int * t, int n)
{
  int i;
  for (i = 0; i < n; i++)
    {
      t[i] = f(t[i]);
    }
}

int
main(void)
{
  int tab[] = {1, 2, 3, 4};
  int (* f)(int) = fois_deux;
  int i;
  appliquer_tableau(f, tab, 4);

  for (i = 0; i < 4; i++)
    {
      printf("%d ", tab[i]);
    }
  printf("\n");

  return EXIT_SUCCESS;
}
```

Exercice 2 Utilisation des pointeurs de fonction dans un algorithme de tri de tableau :

Voici un algorithme de tri des éléménts d'un tableau dans l'ordre croissant :

```
void
tri_croissant(int * t, int n)
{
  int i,i_min,j,tmp;
  for ( i = 0; i < n - 1; i++)
    {
      i_min = i;
      for (j = i; j < n; j++)
        {
          if (t[j] < t[i_min])
            {
              i_min = j;
            }
        }
      tmp = t[i_min];
      t[i_min] = t[i];
      t[i] = tmp;
    }
}
```

Voici un algorithme de tri des éléménts d'un tableau dans l'ordre décroissant :

```
void
tri_decroissant(int * t, int n)
{
  int i,i_max,j,tmp;
  for ( i = 0; i < n - 1; i++)
    {
      i_max = i;
      for (j = i; j < n; j++)
        {
          if (t[j] > t[i_max])
            {
              i_max = j;
            }
        }
      tmp = t[i_max];
      t[i_max] = t[i];
      t[i] = tmp;
    }
}
```

Ces deux algorithme sont identiques, à l'exception de l'opérateur de comparaison. Afin d'éviter de dupliquer du code, nous allons utiliser les pointeurs de fonctions.

- Ecrire une fonction `superieur(int a, int b)` qui renvoie 1 si `a` est supérieur à `b`, 0 s'ils sont égaux et -1 sinon
- Ecrire une fonction `inferieur(int a, int b)` qui renvoie 1 si `a` est inférieur à `b`, 0 s'ils sont égaux et -1 sinon
- Ecrire une fonction `tri`, qui tri un tableau `t` de taille `n` selon une fonction `compare`
- Ecrire une fonction `main` qui teste `tri`

► **Correction**

```
#include <stdio.h>
#include <stdlib.h>

int
superieur(int a, int b)
{
  return (a > b)? 1 : ((a == b)? 0 : -1);
}

int
```

```
inferieur(int a, int b)
{
  return -superieur(a, b);
}

void
tri(int * t, int n, int compare_int(int a, int b))
{
  int i_min,i,j,tmp;
  for (i = 0; i < n - 1; i++)
    {
      i_min = i;
      for (j = i; j < n; j++)
        {
          if (compare_int(t[j],t[i_min]) < 0)
            {
              i_min = j;
            }
        }
      tmp = t[i_min];
      t[i_min] = t[i];
      t[i] = tmp;
    }
}

void
afficher_tableau(int * t, int n)
{
  int i;
  for (i = 0; i < n; i++)
    {
      printf("%d ", t[i]);
    }
  printf("\n");
}


int
main(void)
{
  int t[] = {6, 2, 8, 6, 4, 8, 2, 9, 2, 1};

  int (* compare1)(int,int) = superieur;
  int (* compare2)(int,int) = inferieur;

  tri(t, 10, compare1);
  afficher_tableau(t, 10);

  tri(t, 10, compare2);
  afficher_tableau(t, 10);

  return EXIT_SUCCESS;
}
```

2 Le Débuggeur `gdb`

Exercice 3 Le but de cet exercice est de vous familiariser avec le débuggueur `gdb`. Un débuggueur est un logiciel qui vous permet d'exécuter pas à pas vos programmes (fonction par fonction, instruction par instruction, etc.) et d'afficher les valeurs des variables, afin de mieux comprendre son comportement et de trouver les erreurs de programmation. Pour pouvoir bénéficier de toutes les capacités du débuggueur, vous devrez compiler vos programmes avec l'option `-g`.
Principales fonctionnalités du débuggueur :

– `run` : permet de lancer l'exécution ;
– `step` et `next` : permettent d'exécuter instruction par instruction et fonction par fonction le programme ;
– `break (at)` : place un point d'arrêt
– `cont` : permet de continuer l'exécution jusqu'à la fin du programme, ou jusqu'au point d'arrêt suivant ;
– `print` et `display` : affichent de manière ponctuelle ou en continu la valeur d'une expression (variable,etc.) ;
– `where` : affiche la pile des appels de fonctions ;
– `up` et `down` : permettent de remonter d'un appel de fonction et d'annuler le up.

Vous trouverez à l'url `http://cedric.cnam.fr/~lambe_a1/C.html` une archive contenant quatres programmes. Vous étudierez ces programmes au moyen du débuggueur. Vous êtes censés localiser les erreurs et en voir les manifestations sans modier le code du programme. Vous corrigerez cependant les erreurs trouvées. Vous trouverez plus de détails sur `gdb` ici : `http://www-rocq.inria.fr/secret/Anne.Canteaut/COURS_C/gdb.html`