Programmation C - Autres type de données

Laurent Signac – CC-BY-SA – 07-09-20 1651 bc1abc28fd325cca6054

1 Énumérations

Les énumérations permettent de définir un nouveau type discret, contenant des valeurs telles que des couleurs, des états,...

L'exemple le plus courant est sans doute:

```
// Définition du type enum couleur
enum couleur {rouge,orange,jaune,vert,bleu,violet,indigo};

// Utilisation de variables de ce type
enum couleur c1,c2;

c1=orange;
c2=vert;
```

La réalité est que le compilateur associe un entier, en partant de 0, à chaque valeur possible pour l'énumération. L'utilisateur manipule de son côté des symboles plus évocateurs que des nombres.

Il est toutefois possible de contrôler les entiers utilisés:

```
enum jour {lundi=1,mardi,mercredi,jeudi,vendredi,samedi,dimanche};
```

Dans le cas qui précède, `lundi` sera associé à 1 (plutôt que 0 par défaut), et les autres valeurs suivront. Chaque valeur peut néanmoins être spécifiée:

```
enum etat {sain=0, feu=2, cendre=4};
```

2 Champs de bits

Dans certains cas, il est nécessaire de pouvoir contrôler le nombre de bits exacts sur lesquels une information est stockée (plutôt que de s'accommoder des 8 bits d'un `unsigned char` par exemple).

La déclaration d'un champ de bits ressemble à celle d'une structure. Pour chaque membre de la structure, qui peut être signé ou non signé, on indique combien de bits il faudra utiliser. Ainsi, si on veut diviser un mot de 16 bits en 2 groupes de 6 bits, et un groupe de 4, on peut écrire:

```
struct state {
  int grp1 : 6;
  unsigned int grp2 : 6;
  int val : 4;
};
```

Les trois champs se nommeront alors `grp1`, `grp2` et `val`. Enfin, `grp1` pourra prendre des valeurs entre -32 et +31, `grp2` entre 0 et 63, et `val` entre 0 et 15.

On accède aux données comme on le ferait avec une structure:

```
struct state var;
var.grp2=50;
...
```

3 Unions

Les unions permettent de créer un type de données qui pourra être interprété de différentes manières, c'est à dire décodé selon les principes d'un certain type ou d'un autre.

Voici par exemple un type de donné, nommé `tabchar` qui peut être interprété comme un entier codé sur 4 octets (`int`) ou bien comme un tableau de 4 octets:

```
union tabchar{
  int val;
  unsigned char tab[4];
};
```

On peut ensuite interpréter la donnée de ce type comme un entier ou comme un tableau de `char`:

```
union tabchar v;
int i;
v.val=65538;
for (i=0;i<4;i++) {
  printf("%d ",v.tab[i]);
}
printf("%d\n",v.val);
```

Le programme affichera : 2 0 1 0 65538

(Arrivez-vous à expliquer cet affichage ?)