Les bases de la programmation en C

Par Samir OTMANE

Table des matières

1 Un peu d'histoire

Le C a été conçu en 1972 par Dennis Richie et Ken Thompson, chercheurs aux Bell Labs, afin de développer un système d'exploitation UNIX sur un DEC PDP-11. En 1978, Brian Kernighan et Dennis Richie publient la définition classique du C dans le livre The C Programming language [6]. Le C devenant de plus en plus populaire dans les années 80, plusieurs groupes mirent sur le marché des compilateurs comportant des extensions particulières. En 1983, l'ANSI (American National Standards Institute) décida de normaliser le langage ; ce travail s'acheva en 1989 par la définition de la norme ANSI C. Celle-ci fut reprise telle quelle par l'ISO (International Standards Organization) en 1990. C'est ce standard, ANSI C, qui est décrit dans le présent document.

2 La compilation

Le C est un langage compilé (par opposition aux langages interprétés). Cela signifie qu'un programme C est décrit par un fichier texte, appelé fichier source. Ce fichier n'étant évidemment pas exécutable par le microprocesseur, il faut le traduire en langage machine. Cette opération est effectuée par un programme appelé compilateur. La compilation se décompose en fait en 4 phases successives :

1. Le traitement par le préprocesseur : le fichier source est analysé par le préprocesseur qui effectue des transformations purement textuelles (remplacement de chaînes de caractères, inclusion d'autres fichiers source ...).
2. La compilation : la compilation proprement dite traduit le fichier généré par le préprocesseur en assembleur, c'est-à-dire en une suite d'instructions du microprocesseur qui utilisent des mnémoniques rendant la lecture possible.
3. L'assemblage : cette opération transforme le code assembleur en un fichier binaire, c'est-à-dire en instructions directement compréhensibles par le processeur. Généralement, la compilation et l'assemblage se font dans la foulée, sauf si l'on spécifie explicitement que l'on veut le code assembleur. Le fichier produit par l'assemblage est appelé fichier objet.
4. L'édition de liens : un programme est souvent séparé en plusieurs fichiers source, pour des raisons de clarté mais aussi parce qu'il fait généralement appel à des librairies de fonctions standard déjà écrites. Une fois chaque code source assemblé, il faut donc lier entre eux les différents fichiers objets. L'édition de liens produit alors un fichier dit exécutable.

Les différents types de fichiers utilisés lors de la compilation sont distingués par leur suffixe. Les fichiers source sont suffixés par .c, les fichiers prétraités par le préprocesseur par .i, les fichiers assembleur par .s, et les fichiers objet par .o. Les fichiers objets correspondant aux librairies pré-compilées ont pour suffixe .a.

Le compilateur C sous UNIX s'appelle **cc**. On utilisera de préférence le compilateur **gcc** du projet GNU. Ce compilateur est livré gratuitement avec sa documentation et ses sources. Par défaut, gcc active toutes les étapes de la compilation. On le lance par la commande

gcc [options] fichier.c [-llibrairies]

Par défaut, le fichier exécutable s'appelle **a.out**. Le nom de l'exécutable peut être modifié à l'aide de l'option -o.
Les éventuelles librairies sont déclarées par la chaîne -llibrairie. Dans ce cas, le système recherche le fichier liblibrairie.a dans le répertoire contenant les librairies pré-compilées (généralement /usr/lib/). Par exemple, pour lier le programme avec la librairie mathématique, on spécifie -lm. Le fichier objet correspondant est libm.a. Lorsque les librairies pré-compilées ne se trouvent pas dans le répertoire usuel, on spécifie leur chemin d'accès par l'option -L.

Les options les plus importantes du compilateur gcc sont les suivantes :

-c : supprime l'édition de liens ; produit un fichier objet.
-E : n'active que le préprocesseur (le résultat est envoyé sur la sortie standard).
-g : produit des informations symboliques nécessaires au débogueur.
-Inom-de-répertoire : spécifie le répertoire dans lequel doivent être recherchés les fichiers en-têtes à inclure (en plus du répertoire courant).

-Lnom-de-répertoire : spécifie le répertoire dans lequel doivent être recherchées les librairies précompilées (en plus du répertoire usuel).
-o nom-de-fichier : spécifie le nom du fichier produit. Par défaut, le exécutable fichier s'appelle a.out.
-O, -O1, -O2, -O3 : options d'optimisations. Sans ces options, le but du compilateur est de minimiser le coût de la compilation. En rajoutant l'une de ces options, le compilateur tente de réduire la taille du code exécutable et le temps d'exécution. Les options correspondent à différents niveaux d'optimisation : -O1 (similaire à -O) correspond à une faible optimisation, -O3 à l'optimisation maximale.
-S : n'active que le préprocesseur et le compilateur ; produit un fichier assembleur.
-v : imprime la liste des commandes exécutées par les différentes étapes de la compilation.
-W : imprime des messages d'avertissement (warning) supplémentaires.
-Wall : imprime tous les messages d'avertissement.

3 Les composants élémentaires du C

Un programme en langage C est constitué des six groupes de composants élémentaires suivants :

- les identificateurs,
- les mots-clefs,
- les constantes,
- les chaînes de caractères,
- les opérateurs,
- les signes de ponctuation.

On peut ajouter à ces six groupes les commentaires, qui sont enlevés par le préprocesseur.

3.1 Les identificateurs

Le rôle d'un identificateur est de donner un nom à une entité du programme. Plus précisément, un identificateur peut désigner :

- un nom de variable ou de fonction,
- un type défini par typedef, struct, union ou enum,
- une étiquette.

Un identificateur est une suite de caractères parmi :

- les lettres (minuscules ou majuscules, mais non accentuées),
- les chiffres,
- le ``blanc souligné" (_).

Le premier caractère d'un identificateur ne peut pas être un chiffre. Par exemple, var1, tab_23 ou _deb sont des identificateurs valides ; par contre, 1i et i:j ne le sont pas. Il est cependant déconseillé d'utiliser « _ » comme premier caractère d'un identificateur car il est souvent employé pour définir les variables globales de l'environnement C.

ATTENTION : ***Les majuscules et minuscules sont différenciées.***

Le compilateur peut tronquer les identificateurs au-delà d'une certaine longueur. Cette limite dépend des implémentations, mais elle est toujours supérieure à 31 caractères. (Le standard dit que les identificateurs externes, c'est-à-dire ceux qui sont exportés à l'édition de lien, peuvent être tronqués à 6 caractères, mais tous les compilateurs modernes distinguent au moins 31 caractères).

3.2 Les mots-clefs

Un certain nombre de mots, appelés mots-clefs, sont réservés pour le langage lui-même et ne peuvent pas être utilisés comme identificateurs. L'ANSI C compte 32 mots clefs :

auto	**const**	**double**	**float**	**int**	**short**	**struct**	**unsigned**
break	**continue**	**else**	**for**	**long**	**signed**	**switch**	**void**
case	**default**	**enum**	**goto**	**register**	**sizeof**	**typedef**	**volatile**
char	**do**	**extern**	**if**	**return**	**static**	**union**	**while**

que l'on peut ranger en catégories :

- **les spécificateurs de stockage**
  auto register static extern typedef
- **les spécificateurs de type**
  char double enum float int long short signed struct union unsigned void
- **les qualificateurs de type**
  const volatile
- **les instructions de contrôle**
  break case continue default do else for goto if switch while
- **divers**
  return sizeof

3.3 Les commentaires

Un commentaire débute par **/*** et se termine par ***/**. Par exemple,

```
/*  Ceci est un commentaire */
```

On ne peut pas imbriquer des commentaires. Quand on met en commentaire un morceau de programme, il faut donc veiller à ce que celui-ci ne contienne pas de commentaire.

4 Structure d'un programme C

Une *expression* est une suite de composants élémentaires syntaxiquement correcte, par exemple
x = 0
ou bien

```
(i >= 0) && (i < 10) && (p[i] != 0)
```

Une *instruction* est une expression suivie d'un point-virgule. Le point-virgule signifie en quelque sorte ``évaluer cette expression". Plusieurs instructions peuvent être rassemblées par des accolades { et } pour former une instruction composée ou bloc qui est syntaxiquement équivalent à une instruction. Par exemple,

Ceci est un exemple, cliquez sur le lien de téléchargement pour obtenir le cours complet.