Introduction au langage de programmation procédurale C

Hanen Jabnoun

Introduction

Pourquoi apprendre le langage C ?

Le C a une longue histoire... Créé il y a plus de 40 ans, il a beaucoup évolué depuis. Nous expliquerons ici son histoire, décrirons ses principales caractéristiques et domaines d'utilisation, et fournirons des raisons de l'apprendre (ou de ne pas l'apprendre).

Avec son approche bas-niveau, le C permet d'obtenir des programmes très optimisés, pratiquement autant que s'ils avaient été écrits directement en assembleur. Avec un peu d'effort, il est même possible d'utiliser une approche orientée objet, au prix d'une certaine rigueur que le langage et les compilateurs, dans une certaine mesure, sont très loin d'imposer.

Les systèmes d'exploitation pour ordinateur de bureau les plus répandus actuellement sont Windows de Microsoft, Mac OS X d'Apple, et GNU/Linux. Ils sont tous trois écrits en langage C. Pourquoi ? Parce que les systèmes d'exploitation tournent directement au-dessus du matériel de la machine. Il n'y a pas de couche plus basse pour gérer leurs requêtes. À l'origine, les systèmes d'exploitation étaient écrits en assembleur, ce qui les rendait rapides et performants. Toutefois, écrire un OS en assembleur est une tâche pénible et cela produit du code qui ne peut s'exécuter que sur une seule architecture processeur, comme l'Intel X86 ou l'AMD 64. Écrire un OS dans un langage de plus haut niveau, comme le langage C, permet au programmeur de porter son système d'exploitation sur une autre architecture sans avoir à tout réécrire [1].

Mais pourquoi utiliser le C et non Java, Basic ou Perl ?

Principalement à cause de la gestion de la mémoire. À la différence de la plupart des autres langages de programmation, le langage C permet au programmeur de gérer la mémoire de la manière qu'il aurait choisie s'il avait utilisé l'assembleur. Les langages comme le Java et le Perl permettent au programmeur de ne pas avoir à se soucier de l'allocation de la mémoire et des pointeurs. C'est en général un point positif, car il est assez pénible et inutile de devoir se soucier de la gestion de la mémoire lorsqu'on écrit un programme de haut niveau comme un rapport sur les résultats trimestriels [1].

Cependant lorsqu'on parle d'écrire un programme de bas niveau comme la partie du système d'exploitation qui s'occupe d'envoyer la chaîne d'octets correspondant à notre rapport trimestriel depuis la mémoire de l'ordinateur vers le buffer de la carte réseau afin de l'envoyer vers une imprimante réseau, avoir un accès direct à la mémoire est fondamental — ce qui est impossible à faire dans un langage comme Java par exemple. Les compilateurs C produisent de plus très souvent un code rapide et performant [1].

Est-ce vraiment si merveilleux que le langage C soit un langage si répandu ?

Par effet domino, la génération suivante de programmes suit la tendance de ses ancêtres. Les systèmes d'exploitation écrits en C ont toujours des librairies écrites en C. Ces bibliothèques

système sont à leur tour utilisées pour écrire des bibliothèques de plus haut niveau (comme OpenGL ou GTK) et le programmeur de ces bibliothèques décide souvent d'utiliser le même langage que celui utilisé par ces bibliothèques système. Les développeurs d'applications utilisent ces bibliothèques de haut niveau pour écrire des traitements de texte, des jeux, les lecteurs multimédia, etc... La plupart d'entre eux choisiront d'utiliser pour leur programme le même langage que les bibliothèques de haut niveau. Et le schéma se reproduit à l'infini... [1]

Figure: Kenneth Thompson (à gauche) et Dennis Ritchie (à droite), les créateurs du langage C [1]

Bases de Langage C

Exemple de Programme C :

Voici un premier programme. Il est fonctionnel, même s'il n'est pas normalisé. Il affiche le mot Bonjour à l'écran. À l'aide de votre éditeur de texte (dans la fenêtre Scite donc), tapez le texte qui se trouve à l'intérieur du cadre suivant [2] :

```
main () {
  puts ("Bonjour");
  getchar ();
}
```

Un exemple de fichier en-tête

Vous trouverez ci-dessous, un extrait du fichier en-tête stdio.h. On y retrouve notamment la déclaration de puts (en dernière ligne de l'extrait) que nous venons de mentionner et la déclaration de printf [2] :

```
/* Write formatted output to STREAM. */
extern int fprintf __P ((FILE *__restrict __stream,
      __const char *__restrict __format, ...));
/* Write formatted output to stdout. */
extern int printf __P ((__const char *__restrict __format, ...));
/* Write formatted output to S. */
extern int sprintf __P ((char *__restrict __s,
      __const char *__restrict __format, ...));

/* Write formatted output to S from argument list ARG. */
extern int vfprintf __P ((FILE *__restrict __s,
       __const char *__restrict __format,
       _G_va_list __arg));
/* Write formatted output to stdout from argument list ARG. */
extern int vprintf __P ((__const char *__restrict __format,
      _G_va_list __arg));
/* Write formatted output to S from argument list ARG. */
extern int vsprintf __P ((char *__restrict __s,
       __const char *__restrict __format,
       _G_va_list __arg));

/* Write a string, followed by a newline, to stdout. */
extern int puts __P ((__const char *__s));
```

Squelette de programme

On peut définir le squelette d'un programme C de la façon suivante [2] :

```
/* Déclaration des fichiers d'entêtes de bibliothèques */

int main () {
  /* Déclaration des variables (cf. chapitres suivants...) */

  /* Corps du programme */
  getchar(); /* Facultatif mais permet
                d'attendre l'appui d'une touche */

  return 0; /* Aucune erreur renvoyée */
}
```

Blocs

La partie de programme située entre deux accolades est appelée un bloc. On conseille de prendre l'habitude de faire une tabulation après l'accolade. Puis retirer cette tabulation au niveau de l'accolade fermante du bloc [2]. Ainsi, on obtient :

```
int main () {
  Tabulation
  Tout le code est frappé à cette hauteur
}

Retrait de la tabulation
Tout le texte est maintenant frappé à cette hauteur.
```

Commentaires

Bien commenter un programme signifie qu'une personne ne connaissant pas votre code doit pouvoir le lire et le comprendre. Les commentaires sont indispensables dans tout bon programme. Ils peuvent être placés à n'importe quel endroit [2]. Ils commencent par /* et se terminent par */ :

/ Commentaire */*

Comme nous l'avons déjà mentionné, vous trouverez aussi parfois des commentaires C++ :

// Le reste de la ligne est un commentaire

Exercice d'application

Écrivez un programme qui :

– affiche « Salut toi, appuie sur une touche s'il te plaît » ;
– attend l'appui d'une touche ;
– affiche : « Merci d'avoir appuyé sur une touche ».

Une fois que vous serez satisfait de votre solution, vous pourrez la comparer avec la solution qui apparaît un peu plus loin [2].

Références

[1]Cours Programmation C : https://commons.wikimedia.org/wiki/File:Programmation_C-fr.pdf

[2] Le C en 20 heures : https://framabook.org/docs/c20h/C20H_integrale_creative-commons-by-sa.pdf