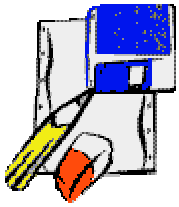
Le langage C.

***The C Programming Language*, Dennis Ritchie et Brian Kernighan.**
Traduction réalisée par JF.Groff et E.Mottier avec la collaboration de E.Allard.
ISBN : 2-225-82070-8



Michel Wathelet Avril 2021

Introduction

Ce langage de programmation conçus par Dennis Ritchie aux Laboratoires Bell AT&T dans les années 70 a servi d'ossature pour réécrire le système d'exploitation UNIX développé dans la même société par Ken Thomson.

C'était un boulot, réécrire un système d'exploitation quasiment développé en langage d'assemblage avec un langage de type algorithmique et qui plus est quand on voit ce qu'il est advenu laisse de l'espoir sur l'évolution de l'intelligence humaine (mais il y a encore du boulot pour l'évolution.)

Cela fait plus de quarante années que je pianote du C sur un clavier, et j'en suis encore aujourd'hui à me demander si je ne devrais pas passer à autre chose. Mais j'ai difficile de concevoir quelque chose d'aussi simple et puissant à utiliser que ce langage qui peut vite devenir addictif. Plus on le cajole, plus on tripatouille, plus de réponses nous reviennent, et plus on voit la beauté de l'écriture algorithmique se dessiner, prendre forme et s'épanouir dans une séquence qui aussi longue soit-elle dégage la beauté du raisonnement. Que l'auteur de ce langage en soit remercié à jamais. Il nous à donné un outil universel qui devrait faire partie intégrante des cours obligatoires dans toutes les écoles de la planète.

Il existe une documentation très importante sur le langage C'. Mais l'utilisateur de ce langage doit, avoir, je pense sous la main au moins ce livre clé.

***The C Programming Language*, « The White Book », écrit par Dennis Ritchie, créateur du langage, et Brian Kernighan.**

Je travaille sur la traduction réalisée par JF.Groff et E.Mottier avec la collaboration de E.Allard.
ISBN : 2-225-82070-8

Je ne vais pas réécrire ce qui existe déjà, mais voyons plutôt notre environnement de travail.

Les outils et surtout leur diffusion ont bien changé en 50 ans. Aujourd'hui, avec les outils GNU, bien des choses impossibles sont devenues réalisables par exemple Linux pour ne citer que cette importante réalisation logicielle, mais aussi la série des compilateurs sans oublier Emacs. On peut dire qu'à ce jour pour le jeune informaticien, les anciens ont mis dans son berceau un cadeau dont il ne peut mesurer l'importance. Plus tard, peut-être lorsqu'il aura pris de l'âge, il se rendra compte de l'immense contribution du mouvement du logiciel libre dans sa corbeille.

- ✓ Le langage C'
- ✓ Le compilateur C du programme GNU
- ✓ L'éditeur Emacs

Ces trois outils remarquables vous en avez accès aux code source et de plus une communauté toujours prête a vous aider.

On pourra y adjoindre plus tard des primitives de base de données ou de gestionnaire de fichiers.

Les ordinateurs actuels sont peu lourds et puissants et sans parler du réseau Internet, un vrai bonheur. Dans lequel le monde s'est engouffré. Qu'est-ce que cela à avoir avec ce langage informatique me dire-vous.

Le langage C, est idéal pour naviguer dans tout cela. Que le code soit de haut ou de bas niveau, que l'on ait besoin d'une approche plus technique de gestion d'interruptions, de manipuler le processeur dans ses recoins les plus inattendus ou de travailler sur des structures parallèles voire même de créer du code autopropageable. N'oublions pas non plus la gestion des communications.

Le langage C, est non seulement un outil formidable, mais je ne connais pas de langage qui puisse se mettra en toute circonstance au niveau de son utilisateur.

Un autre atout et non des moindres est la possibilité d'avoir a sa disposition un système d'exploitation disponible en sources et dont le noyau est écrit par son concepteur Linus Torvald en langage C.

Chap I.

Le C est un langage de mécanicien, nous avons une base succincte de quelques lignes nécessaires et suffisantes pour démarrer le moteur et puis l'ajusteur qui sommeille en vous peut transformer la poussette en formule 1.

Préférences des outils:

Même un vieux pc des années 1980 est bon pour cette étude et une vieille imprimante à ruban peut faire l'affaire ce n'est que pour lister du code. Pour les communications, si vous avez un modem intégrer, c'est bon, sinon ce n'est pas très difficile à trouver.

Un système d'exploitation Linux, clone d'Unix, gratuit, puissant et surtout un système alliant souplesse et performance en constante augmentation. Moi, je suis plutôt fan de la distribution Debian, mais vous avez le choix entre plusieurs distributions qui sont disponibles soit en CD, soit sur le réseau Internet.

Ensuite les outils du projet GNU.
Richard Stallman est l'instigateur de ce projet, après avoir créer *emacs* un éditeur puissant il continua sur sa lancée pour créer le projet GNU, un ensemble d'outils de développement. A ce jours l'informaticien peut trouver des outils couvrant la majeure partie des différents domaines de son métier. Enorme avantage sur les vieux comme moi qui ont connu les cartes perforées.

Pour faire simple, le livre de base de Kernighan et Ritchie, un ordinateur équipé avec un système linux et les outils de développements GNU que sont :

- ✓ Le compilateur gcc
- ✓ Son débogueur
- ✓ L'outil de gestion du code make
- ✓ Les bibliothèques
- ✓ L'éditeur emacs ou votre éditeur préféré.

Le premier chapitre est une présentation générale du langage, on voit les principales possibilités d'écriture

L'écriture.
Je ne saurais pas trop, vous recommandez de lire et de vous imprégner en même temps de la norme ANSI du langage. Sur le site vous pouvez la regarder ou simplement la télécharger

ANSSI-PA-073
Version 1.1 - 29/05/2020
Licence ouverte/Open Licence (Étalab - v1)

https://www.ssi.gouv.fr/guide/regles-de-programmation-pour-le-developpement-securise-de-logiciels-en-langage-c/
.
AGENCE NATIONALE DE LA SÉCURITÉ DES SYSTÈMES D'INFORMATION
.
ANSSI - 51, boulevard de La Tour-Maubourg, 75700 PARIS 07 SP
www.ssi.gouv.fr / conseil.technique@ssi.gouv.fr

Sur ce site vous trouverez de la documentation intéressante .

Écrire entraîne automatiquement une documentation. En faire trop alourdit le code, mais ne rien faire peut être plus désastreux encore.

Un programme C à ceci de particulier que nous nous trouvons dans un environnement de fonctions. La fonction main() obligatoire à toute application écrite en C est la seule nécessaire au bon fonctionnement du programme..

L'environnement de C comporte un préprocesseur utile à bien des égards. Pour que la commande puisse accéder au préprocesseur, elle commence par le caractère #. Étudiez votre compilateur (CC – GNU) vous y trouverez toutes les commandes de compilations conditionnelles entre autres.

Celles qui sont plus intéressantes sont :

#Define
#Include

#if 0
#endif

#Define, on peut la considérer comme une macro-instruction, destinée uniquement au préprocesseur, elle autorise un remplacement de valeur. Il faudra faire très attention avec cette définition qui peut si on n'y prend pas garde vous faire faire des erreurs.

#include permet d'effectuer des inclusions dans votre code.

Documenter entraîne l'utilisation des signes /* commentaire….*/ tout ce qui se trouve entre ces deux caractères est ignoré par le compilateur. Dans la première période du C, c'étaient les seuls délimiteurs pour une ligne de documentation. Maintenant, on peut comme pour le C++ se servir des deux barres obliques en début de ligne : //commentaire.

Une autre façon bien plus simple et nettement plus portable est d'utiliser le couple ::

#if 0
  commentaire ….
  Commentaire…..
  …..
#endif

Le préprocesseur est dans tous les cas d'une utilité dont il faut savoir tirer parti. On peu pratiquement considérer qu'il est identique à un macro-assembleur et qu'il peut s'assimiler de fait et être utiliser comme un langage proche de la machine sur lequel il est employer. Parmi ces possibilités on peut citer le pilotage des erreurs dans le programme ou le système sous-jacent. Le préprocesseur fera donc l'objet d'une étude particulière de votre part.

L'ossature d'une application en C prend l'allure suivante :

```
#define
#include
main()
{
  appel de fonction(s);
]
```

et c'est bon. Plus simple que cela, n'est pas évident.

Tester le premier programme en C du livre et modifier le de façon à utiliser define. Trouvez plusieurs solutions intelligentes et pertinentes.

Avant de commencer à lire et à coder, maîtriser votre compilateur et ses outils annexe – make, emacs par exemple et savoir bien se servir de votre distribution linux. Vous pouvez pour cela tester les programmes su 1er chapître de K&R.

Si votre compilation se passe sans problème, le démarrage est bon et cela est de bon augure.

J'utilise la distribution GNU/LINUX de Debian ver.10
Et les outils présnt dans cette distribution. – emacs, gcc… sur un ordi,ateur portable de marque HP.

Le premier chapitre traite de la présentation générale. Quelques lignes de code pour étayer le texte. Premières fonctions comme main() et printf(). La présentation des variables et les constantes symboliques. Bref un aperçu de la syntaxe et des objets traités dans ce langage.

Vous trouverez également quelques exercices, soit dans le corps du chapitre,soit en fin de chapitre. Le " Kernighan " contient les énoncés d'une centaine d'exercices plus ou moins compliqués mais sans leur solution. **Un livre complémentaire de corrections des exercices du "Kernighan".** Des auteurs C. Tondo et S. Gimpel, reprend tous ces énoncés et en fournit une solution détaillée.

Chaque solution a été rédigée en n'ayant recours qu'aux connaissances acquises au moment où l'exercice est posé, de manière à suivre la même progression que le Kernighan. C'est donc une démarche intelligente et qui favorise la compréhension du texte.

C'est donc un bon achat. Vous pourrez ainsi voir un exemple de solution du problème, et éventuellement le modifier. Mais tout aussi important vous mettre en confiance dans l'utilisation du compilateur et de ses outils annexes.

Le langage C est typé, en réalité le C comporte deux types, l'entier et le réel. Les autres types ne sont en fait que des variantes des entiers :

- ✓ **Les caractères (char)**
- ✓ **Les entiers courts (short int)**
- ✓ **Les entiers longs (long int)**
- ✓ **Les entiers normaux (int)**

Ces quatre variantes pouvant en plus être signée ou non (signed ou unsigned).

De plus le langage C possède trois variantes des réels :

- ✓ **Le réel en simple précision (float)**
- ✓ **Le réel en double précision (double)**
- ✓ **Le réel en quadruple précision (long double)**

Vous avez compris bien sûr qu'il s'agît d'une réservation de place mémoire. Par définition l'octet sera considéré comme étant la plus petite unité de stockage pour conserver une donnée de base. Vous ne manquerez pas également de connaître le type de machines utilisées 8, 16, 32,…. Bits.

En langage C, le caractère, est une variante du nombre entier . On peut les manipuler dans le sens arithmétique exactement comme de simple nombre. Le problème pour vous sera toujours de savoir si le caractère est signé ou non.

```
/* Taille d'un octet
   pgm tailleoctet.c
-------------------------------------------------------------------------------------------*/

#include<stdio.h>

int main()
{
  printf(" Sur cette machine la taille de l'octet est de  : %d\n, sizeof(char));
}
/* fin tailleoctet.c ----------------------------------------------------------------------*/
```

compilation
cc tailleoctet.c –o tailleoctet

test du programme ./tailleoctet <Ret>
Vous pouver faire cela sans quitter émacs. Par contre, il faut impérativement ajouter le fichier **<stdio.h>**

Le système retourne la valeur 1

La fonction *main* ne fait pas partie de la bibliothèque standard, elle sert de point d'entrée dans le programme et n'est appelée qu'une seule fois.

LES INSTRUCTIONS

Les objets du langage C que sont les constantes, variables et fonctions sont combinées afin de former des phrases, des expressions compréhensibles et utilisables par le processeur. La combinaison passe alors par la brique nécessaire et suffisante qu'est l'instruction.

Sa forme peut être une déclaration, une affectation ou même un appel de fonction. C'est en fait une chaîne valide du point de vue de la syntaxe et terminée obligatoirement par un point-virgule.

L'instruction est la base et l'ensemble de vos instructions forme votre programme.

```
printf("Bonjour,le monde");

Total = 1;
```

sont des instructions. Vous aurez ainsi des instructions élémentaires comme les deux exemples ci-dessus, mais vous pouvez avoir une instruction composée ou une instruction d'échappement.

VARIABLES

Pour stocker des informations, nous avons besoin de tiroirs, Une variable joue ce rôle dans notre programme. Ainsi l'écriture :

```
Int tiroir ;
```
déclaration d'une variable appelée tiroir et dans lequel je stocke des valeurs entières Dans ce cas elle n'est ici pas initialisée et telle qu'elle peut contenir n'importe quoi comme par exemple :

```
Int tiroir ;
Printf(" la variable tiroir contient : %d\n", tiroir);
```

la variable tiroir contient : 502534.

L'initialisation de **int tiroir = 100;** provoque l'affichage corrigé de **la variable tiroir contient : 100**

Vous devrez faire attention au typage et à toujours initialisé vos variables à zéro afin d'être assuré du bon retour de l'information, une variable non initialisée contiendra toujours la valeur stockée à l'emplacement mémoire utilisé par la variable.

Int tiroir = 0; est donc une bonne pratique.

LES TYPES.