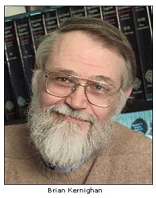
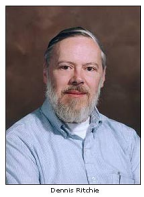
Séance 1 : Introduction au langage C

Un peu d'historique

Le langage C a été inventé dans les années 70 par D. Ritchie et B. W. Kernighan, deux chercheurs du Bell Labs (centre de recherche et de développement d'Alcatel-Lucent depuis 2006).


Brian Kernighan


Dennis Ritchie

Le langage C a été créé à peu près au même moment que le système d'exploitation UNIX. D'ailleurs tous les systèmes d'exploitation modernes comme les différentes versions de Windows, SunOs l'Unix de Sun Microsystems ou Linux, l'Unix libre créé par Linus Torvalds ont été écrits dans ce langage.


Linus Torvalds

Le langage C est donc un langage de base dans toute la recherche et l'industrie informatique moderne et cela ne doit rien au hasard. En effet ce langage présente de nombreux avantages. Tout d'abord, c'est un langage portable, tout programme écrit en C peut être utilisé sur la majorité des systèmes informatiques existants. Ensuite c'est un langage simple, il ne comporte que peu de règles et d'instructions. Pour finir, c'est un langage proche du processeur, les programmes en C sont proches de l'optimum en terme d'utilisation de celui-ci.

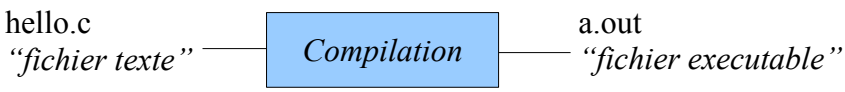
Premier programme en C

Voici le premier programme en C que nous allons écrire, compiler et exécuter ensemble. Tapez le texte suivant dans un éditeur de texte :

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

int main(void)
{
      printf( "Boujour tout le monde!\n");
      return(EXIT_SUCCESS);
}
```

Le seul langage informatique compréhensible par le processeur d'un ordinateur est le langage machine (en général spécifique à chaque type de processeur). Pour tous les autres langages informatiques il faudra donc réaliser une traduction. Dans le cas du langage C, on parle de langage compilé.



La traduction est réalisée une fois pour toutes par un programme appelé compilateur, le programme une fois compilé pourra être exécuté de multiples fois. Pour d'autres langages, on parle de langage interprété, la traduction est réalisée simultanément à l'exécution.

On comprend donc que les langages compilés sont plus rapides à l'exécution que les langages interprétés puisque l'étape de traduction a été effectuée une fois pour toutes lors d'une étape précédente.

Sauvez le programme que vous avez tapé dans un fichier «`hello.c`». Pour réaliser la compilation, ouvrez une fenêtre de commande et tapez l'instruction suivante :

```
gcc hello.c
```

Cette instruction appelle le compilateur (`gcc`) avec comme argument le nom du fichier texte dans lequel nous avons stocké notre premier programme C. Il va générer un fichier en langage machine appelé «`a.out`» qui contient notre programme traduit en langage machine. Si jamais vous n'avez pas correctement tapé le programme le compilateur va vous affirmer que vous avez commis des erreurs et va vous indiquer lesquelles.

Le programme «`a.out`» est donc directement exécutable par l'ordinateur. Pour le vérifier il vous suffit de taper dans la même fenêtre de commande l'instruction suivante :

```
a.out
```

On peut aussi préciser le nom du programme exécutable (au lieu de `a.out`) en utilisant dans `gcc` , l'option `-o` :

```
gcc hello.c -o hello
```

L'exécutable aura alors pour nom `hello` (et non `hello.c` !)

Un peu de décorticage…

Nous allons maintenant décrypter une à une toute les lignes de notre programme.

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

Le langage C permet l'utilisation d'un certain nombre de fonctions. Ces fonctions sont regroupées au sein de librairies. Lorsque l'on utilise une fonction d'une certaine librairie on doit le dire au compilateur par l'intermédiaire de la directive de compilation «`include`». Dans notre programme, nous utilisons la fonction «`printf`» qui fait partie de la librairie standard d'entrée/sortie `stdio.h` et `EXIT_SUCCESS` qui est défini dans `stdio.h`.

```
int main(void)
{
```

Ceci est la déclaration d'une fonction appelée «`main`».
Tout programme en langage C comprend une et une seule fonction «`main`».
Lors de l'exécution du programme c'est cette fonction qui est appelée.
La déclaration d'une fonction se fait par le nom de la fonction précédée de son type, la déclaration des arguments, puis le corps de la fonction.
La fonction `main` ici ne comprend aucun argument, leur déclaration se fait alors par une parenthèse ouvrante suivie de `void` puis d'une parenthèse fermante.

Le corps de la fonction est constituée par un bloc, un bloc étant délimité par une accolade ouvrante et une accolade fermante. `main` est précédée par `int` , ce qui signifie que la fonction `main` est du «type `int`» : elle « renverra » un entier par le biais de l'instruction `return(EXIT_SUCCESS)` que nous étudierons ultérieurement.

```
printf( "Boujour tout le monde!\n");
```

Dans cette ligne, on fait appelle à la fonction de sortie standard «`printf`». Cette fonction fait partie de la librairie standard d'entrée/sortie. Elle permet d'afficher une chaîne de caractères à l'écran. Cette chaîne de caractères est passée en argument à la fonction (entre parenthèses).
« `\n` » permet de retourner à la ligne.

Finalement, l'accolade fermante termine le bloc constituant la fonction `main`.

Exercices plus élaboré (à saisir et à compiler)

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

/* Exemple 1 */
/*  Calcul de la surface d'un cercle de rayon donné */

int main(void)
{

      /* Declaration des variables */
      float rayon, aire ;

      /* Saisie du rayon */
      printf("Entrez le rayon : ") ;
      scanf("%f", &rayon) ;

      /* Calcul de la surface */
      aire=3.14159 * rayon * rayon ;

      /* Affichage du résultat */
      printf("La surface du cercle vaut : %f\n", aire);

      return(EXIT_SUCCESS);

}
```

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

/* Exemple 2 */
/*  Calcul de la surface d'un cercle de rayon donné  */
/*  en entrant le rayon comme parametre du programme */
int main(int argc, char* argv[])
{

      /* Declaration des variables */
      float rayon, aire ;

      /* Recuperation du rayon */
      rayon = atof(argv[1]);

      /* Calcul de la surface */
      aire=3.14159 * rayon * rayon ;

      /* Affichage du résultat */
      printf("La surface du cercle vaut : %f\n", aire);

      return(EXIT_SUCCESS);

}
```