Objectif 01
Introduction au langage C

Aziz Salah
salah.aziz@uqam.ca

Département d'informatique
UQÀM

Automne 2013

Plan

Plan

Historique du langage C

- C K&R (1978) publié par Brian W. Kernighan et Denis M. Ritchie
- C ANSI ou C89 aussi C90 ISO : comprend le C K&R et quelques changements du format en plus des prototypes de fonction format C++
- C99 compatible avec C90 avec quelques restrictions et ajoute certains nouveauté comme les fonctions `inline`
- C11 : principalement permet d'améliorer la compatibilité avec le C++ mais ajoute plusieurs nouvelles aspects comme les opérations atomiques et autres

Caractéristiques du langage C

- Simple : syntaxe composée d'un petit ensemble de vocabulaire, facile à apprendre et riche
- Efficace : utilisation des pointeurs, code compact, manipulations proches de l'assembleur
- Portable : facilement portable à d'autres systèmes
- Puissant : utilisé pour la programmation système, le calcul scientifique, les compilateurs ...
- Modulaire et structuré : programme sous forme de modules, code source sous forme de blocs

Caractéristiques du langage C (suite)

- Impératif : un programme est décrit par une séquence d'instructions à exécuter
- Compilé : Un exécutable est généré par le compilateur à partir du code source du programme

Plan

Un premier programme C

Listing 1 – premier_prog.c

```c
/**
 * Un premier programme
 */
#include <stdio.h> // Instructions du préprocesseur
#include <stdlib.h>

int
main()
{
  int annee = 2013; //annee est une variable de type int

  printf("Bonjour!\n");
  printf("Nous sommes en %d!\n", annee);
  return EXIT_SUCCESS; // EXIT_SUCCESS représente 0
}
```

Compilation et exécution d'un programme C

Après édition du programme sous forme du fichier `premier_prog.c` en format `UTF-8`

Au `prompt` à la `ligne de command` sur le `terminal`

```
$ ls
premier_prog.c
$
```

- La commande `ls` permet de lister les fichiers du répertoire courant
- Attention `Unix` distingue entre les lettres majuscules et les minuscules

Compilation et exécution d'un programme C (suite)

Commande de compilation au terminal

```
$ gcc -Wall -std=c99 premier_prog.c -o premier_prog
$ ls
premier_prog.c  premier_prog
$
```

- `gcc` est le compilateur utilisé
- `-Wall, -std=c99` et `-o` sont des options de la commande `gcc`
- `-Wall` dit au compilateur d'afficher tous les avertissements
- `-std=c99` dit au compilateur d'utiliser la norme C99
- `-o` dit au compilateur d'appeler l'exécutable par le nom qui suit (ici c'est `premier_prog`)
- Si le programme comporte des erreurs de compilation, aucun exécutable n'est généré !

Compilation et exécution d'un programme C (suite)

Exécution du programme au terminal

```
$ ./premier_prog
Bonjour!
Nous sommes en 2013!
$
```

- Sans le préfixe "`./`" le système risque de ne pas localiser le fichier exécutable
- L'affichage du programme se fait sur le terminal

Compilation et exécution d'un programme C (suite)

Suppression d'un fichier au terminal

```
$ ls
premier_prog.c premier_prog
$ rm premier_prog
$ ls
premier_prog.c
$
```

- La commande `rm` permet de supprimer un fichier de manière permanente
- Pas de corbeille avec `rm` ☹

Compilation et exécution d'un programme C (suite)

Exécutable par défaut

```
$ gcc -Wall -std=c99 premier_prog.c
$ ls
a.out premier_prog.c
$ ./a.out
Bonjour!
Nous sommes en 2013!
$
```

- Sans l'option `-o`, par défaut l'exécutable s'appelle `a.out`

Plan

Un programme C avec une fonction I

Listing 2 – deuxieme_prog.c

```
1  /**
2  * Un deuxième programme
3  */
4  #include <stdio.h>
5  #include <stdlib.h>
6
7  int maximum (int a, int b); //Déclaration de la fonction
8
9  int
10 main()
11 {
12     int n1;
13     int n2;
14     int resultat;
15
16     printf("Donnez deux entiers séparés par des espaces\n");
```

Un programme C avec une fonction II

```
17      scanf("%d %d",&n1,&n2);
18      resultat = maximum(n1,n2);
19      printf("Le max est %d!\n", resultat);
20
21      return EXIT_SUCCESS; // EXIT_SUCCESS représente 0
22  }
23  int
24  maximum (int a, int b) //Définition de la fonction
25  {
26      int m;
27
28      if ( a < b )
29          m = b;
30      else
31          m = a;
32
33      return m;
34  }
```

Exécution du programme : appelle de `main()`

Appel du `main()`

Appel de printf()

Affichage de "Donnez deux entiers séparés par des espaces\n"

Appel de scanf()

Lecture des deux entiers

Appel de `maximum()`

retourne le max

La valeur retournée par `maximum()` est assignée à `resultat`

Appel de printf

Affichage de "Le max est " ...

retourne `EXIT_SUCCESS`

Compilation et exécution

Au terminal

```
$ gcc -Wall -std=c99 deuxieme_prog.c -o prog
$ ./prog
Donnez deux entiers séparés par des espaces
12 34
Le max est 34!
$
```

- Les caractères blancs (espaces, tabulations, fin de ligne, ...) entre les deux entiers seront ignorés lors de la saisie