Le langage C

François Trahay

Présentation du module

Objectifs du module:

- Maîtriser le langage C
- Savoir s’adresser au système d’exploitation depuis un programme

Modalités:

- Un peu de théorie
- Beaucoup de pratique

Contenu du module

Partie *Programmation*

- CI 1 – Le langage C
- CI 2 – Les types composés / qu'est-ce qu'une adresse ?
  - Exercice Hors-Présentiel
- CI 3 – Faire des programmes modulaires en C
- CI 4 – Les pointeurs
- CI 5 – Debugger un programme

Partie *Système*

- CI 6 – Les fichiers
- CI 7 – Les processus
- CI 8 – Appels système et Sémaphores
- CI 9 – Signaux

Evaluation

- CI 10 – Exercice de synthèse
- CF1 (sur papier) – questions sur l'exercice de synthèse

Déroulement d'une séance

Système de *classe inversée*. Pour chaque séance :

- **Avant** la séance
  - Etude de la partie cours (y compris les commentaires des slides) de la séance à venir
- **Pendant** la séance:
  - Mini-évaluation de la partie cours (Kahoot!)
  - Explications sur les points mal compris
  - Travaux pratiques : expérimentations sur les concepts vus en cours

Attention ! Cela ne fonctionne que si vous travaillez sérieusement **avant** la séance.
Hypothèse: les étudiants suivant ce cours sont des adultes responsables.

Ressources disponibles

Pour vous aider, vous avez à votre disposition:

- Le poly contenant l'ensemble des transparents commentés
- Les transparents en version pdf/html
    - en html, appuyez sur `s` pour afficher les commentaires
- La documentation des fonctions C standard (`man 2 <fonction>` ou `man 3 <fonction>`)
- Une équipe enseignante de choc !

Conseils pour la suite de votre scolarité

- N’utilisez pas d’IA pour vos TP/DM
  - ChatGPT / Copilot sont des outils pour **améliorer la productivité** des développeurs
  - Les TPs servent à **apprendre**, pas à produire
- Ne consultez les corrigés des TP qu’en dernier recours
  - Essayez de faire les exercices
  - Posez des questions aux enseignants

C vs. Java

- langage de *bas niveau* vs. *haut niveau*
- En C, manipulation de la mémoire et de ressources proches du matériel
- *“Un grand pouvoir implique de grandes responsabilités”*[1]
- programmation impérative vs. programmation objet

[1] *B. Parker, Amazing Fantasy, 1962*

Mon premier programme en C

Fichier ***.c**

```c
/* hello_world.c */
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv) {
  printf("Hello World!\n");
  return EXIT_SUCCESS;
}
```

Compilation/execution:

```
$ gcc hello_world.c -o hello_world -Wall -Werror
$ ./hello_world
Hello World!
```

Déclaration de variable

Pour les types simples, déclaration identique à Java:

```
int var1;
int var2, var3, var4;
int var5 = 42;
```

Types disponibles:

- pour les entiers: `int`, `short`, `long`, `long long`
- pour les flottants: `float`, `double`
- pour les caractères: `char`

Opérateurs et Expressions

La liste des opérateurs disponibles est à peu près la même qu'en Java:

- arithmétique : `+`, `-`, `*`, `/`, `%`
- affectation : `=`, `+=`, `-=`, `*=`, `/=`, `%=`
- incrémentation/décrémentation: `++`, `--`
- comparaison: `<`, `<=`, `>`, `>=`, `==`, `!=`
- logique: `!`, `&&`, `||`

Mais également:

- `sizeof n`: donne le nombre d'octets qui constitue une variable/un type `n`

Opérateurs bit à bit

Possibilité de travailler sur des *champs de bits*.

- Opération sur les bits d'une variable
- décalage: <<, >>
- OR : |, AND :&, XOR : ^, NOT : ~
- affectation: <<=, >>=, |=, &=, ^=, ~=

Structures algorithmiques

Comme en Java:

- `for(i=0; i<n; i++) { ... }`
- `while(cond) {... }`
- `do { ... } while(cond);`
- `if (cond) { ... } else { ... }`
- `break` pour sortir d'une boucle

Affichage / Lecture

- Pour afficher: `printf("%d exemple de %f format \n", v1, v2);`
- Pour lire: `scanf("%d-%f", &v1, &v2);`

Fonctions

Déclaration:

```
type_retour nom_fonc(type_param1 param1, type_param2 param2) {
/* déclaration des variables locales */
/* instructions à exécuter */
}
```