Architecture des ordinateurs
(X31I050)

Frédéric Goualard

Laboratoire d'Informatique de Nantes-Atlantique, UMR CNRS 6241
Bureau 112, bât. 11
Frederic.Goualard@univ-nantes.fr

Aspects pratiques 1

- *Syllabus* : description de tous les aspects pratiques (**lisez-le !**)
- Ressources et planning sur madoc
- *Syllabus* : description de tous les aspects pratiques (**lisez-le !**)
- Fascicules présentant les notions vues en cours
- *Syllabus* : description de tous les aspects pratiques (**lisez-le !**)

Aspects pratiques 2

- 10 cours magistraux d'1h20
  - Présentation des points à étudier
  - 7 quiz sur madoc pour la compréhension de chaque partie
  - Fascicules reprenant les notions à connaître
- 12 séances de travaux dirigés d'1h20
  - Faire les exercices de TD **avant** (pas de distribution des feuilles de TD)
  - Mini cours interactif sur les points difficiles relevés
  - 2 évaluations sur table des parties précédentes
- 9 séances de travaux pratiques
  - C/C++
  - Circuits logiques (logisim)
  - Assembleur MIPS (MARS)

Travail personnel **absolument** indispensable en amont des séances

LS2N Planning

Architecture des ordinateurs

2023/2024

Tous groupes

Attention: le planning est susceptible de changer au cours du semestre et de varier en fonction des groupes.

v. 1, 2023-09-08 *10 CMs, 12 TDs, 9 TPs*	**Travail à faire** **(Exercices de TD)**				
(37) 11/09 – 15/09		CM1 Représentation de l'information			
(38) 18/09 – 22/09	1.1 à 1.7 (rappels) 1.9, 1.10, 1.12, 1.14, 1.15	CM2 Représentation de l'information	TD 1.1 Représentation de l'information		
(39) 25/09 – 29/09		CM3 Représentation de l'information Performances	TD 1.2 Représentation de l'information	TP 1.1 Représentation de l'information	
(40) 02/10 – 06/10		CM4 Logique & circuits combinatoires	TD 2 Performances	TP 1.2 Représentation de l'information	
(41) 09/10 – 13/10		CM5 Circuits séquentiels	TD 3.1 Circuits combinatoires	TP 2 Circuits combinatoires	
(42) 16/10 – 20/10		CM6 Circuits séquentiels	CC 1 Rep. Info + Performances TD 3.2 Circuits combinatoires	TP 3.1 Circuits séquentiels	
(43) 23/10 – 27/10			TD 4.1 Circuits séquentiels	TP 3.2 Circuits séquentiels	
(44) 30/10 – 03/11	Distribution du sujet de projet				
(45) 06/11 – 10/11		CM7 Assembleur MIPS	TD 4.2 Circuits séquentiels	TP Mini-projet	
(46) 13/11 – 17/11		CM8 Assembleur MIPS Instructions et chemins de données	TD 5.1 Assembleur MIPS	TP 4.1 Assembleur MIPS	
(47) 20/11 – 24/11		CM9 Instructions et chemins de données	TD 5.2 Assembleur MIPS	TP 4.2 Assembleur MIPS	
(48) 27/11 – 01/12		CM10 Pipelines Caches	CC 2 Circuits combinatoires & séquentiels TD 5.3 Assembleur MIPS	TP CCTP MIPS	
(49) 04/12 – 08/12			TD 6 Instructions & chemins de données		
(50) 11/12 – 15/12			TD 7 Pipelines et caches		

Objectifs du cours et évaluation

- Objectifs du cours
  - Connaître la représentation interne des données/instructions en binaire
  - Comprendre les mécanismes de fonctionnement d'un processeur
  - Comprendre les interactions entre processeur et périphériques (mémoire, unités de stockage de masse, E/S,...)
  - Appréhender l'impact de l'architecture d'un ordinateur sur les performances de programmes écrits dans des langages de haut niveau
- Évaluation

Contrôle continu	2 contrôles « sur table »	25%	50%
	Exercice de TP	10%	
	Projet de TP	15%	
Examen			50%

- Gestion des absences
  - Voir le syllabus

Motivations (1)

Pourquoi un module sur l'architecture des ordinateurs ?

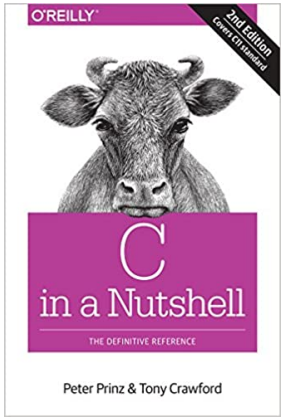
Exemple de la boîte de vitesses d'une voiture : si on connaît son principe de fonctionnement, on sait mieux utiliser le véhicule :

- Prise directe
- Démarrage en seconde sur glace

Motivation (2)

C in a Nutshell, 2e édition :

```
long double factorial(unsigned int n)
{
        long double f = 1;
        while (n > 1) {
                f *= n--;
        }
        return f;
}
```

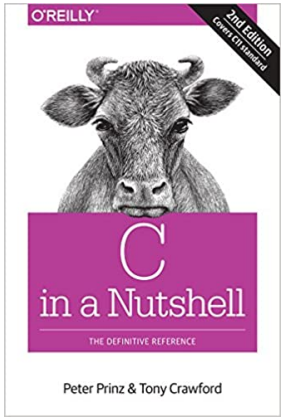


*« Although the factorial of an integer is always an integer, the function uses the type **long double** in order to accommodate very large results. »*

Motivation (2)

C in a Nutshell, 2e édition :

```
long double factorial(unsigned int n)
{
        long double f = 1;
        while (n > 1) {
                f *= n--;
        }
        return f;
}
```



> *« Although the factorial of an integer is always an integer, the function uses the type **long double** in order to accommodate very large results. »*

Problème avec `long double` :

- Partie fractionnaire de 64 bits
- Au-delà de 2^{65} : tous les entiers ne sont pas représentables
- Type `uint64_t` : calcul jusqu'à $n = 20$
- Type `long double` : calcul jusqu'à $n = 25$ (`factorial(26)` faux)

Motivations (3)

```cpp
#include <iostream>

using namespace std;

int main(void) {
  unsigned int matSZ;
  cout << "Taille de la matrice ? ";
  cin >> matSZ;
  int *mat = new int[matSZ*matSZ*sizeof(int)];
  unsigned int ligne, colonne;
  cout << "Position de l'élément ? ";
  cin >> ligne >> colonne;
  int val;
  cout << "Valeur de l'élément ? ";
  cin >> val;
  if (ligne >= matSZ || colonne >= matSZ) {
    cout << "Position invalide" << endl;
  } else {
    mat[ligne*matSZ+colonne] = val;
  }
}
```

```java
import java.util.Scanner;

public class scan {
  public static void main(String[] args)
{
    Scanner in = new Scanner(System.in);
    System.out.printf("Enter i: ");
    int i = in.nextInt();
    int[] M = new int[i*i*4];
    M[5] = 3;
  }
}
```

Motivations (3)

```cpp
#include <iostream>

using namespace std;

int main(void) {
  unsigned int matSZ;
  cout << "Taille de la matrice ? ";
  cin >> matSZ;
  int *mat = new int[matSZ*matSZ*sizeof(int)];
  unsigned int ligne, colonne;
  cout << "Position de l'élément ? ";
  cin >> ligne >> colonne;
  int val;
  cout << "Valeur de l'élément ? ";
  cin >> val;
  if (ligne >= matSZ || colonne >= matSZ) {
    cout << "Position invalide" << endl;
  } else {
    mat[ligne*matSZ+colonne] = val;
  }
}
```

```java
import java.util.Scanner;

public class scan {
  public static void main(String[] args)
{
    Scanner in = new Scanner(System.in);
    System.out.printf("Enter i: ");
    int i = in.nextInt();
    int[] M = new int[i*i*4];
    M[5] = 3;
  }
}
```

- Si `matSZ`==2^{31}==`2147483648` , allocation de 0 octet dans le tas (`new int[size_t]` et `sizeof(size_t)==8`)
- Écrasement possible d'autres variables dans le tas, même avec une saisie contrôlée (exception en Java)

Motivation (4)

```
#include <iostream>
#include <cstring>

using namespace std;

int main(void)
{
  struct {
         char mot_de_passe[15];
         bool mdp_ok = false;
  } data;

  cout << "Mot de passe ? ";
  cin >> data.mot_de_passe;


  if (!strcmp(data.mot_de_passe, "Sesame")) {
    data.mdp_ok = true;
  }

  // [...]

  if (data.mdp_ok) {
    cout << "Mot de passe correct" << endl;
  } else {
    cout << "Mot de passe incorrect" << endl;
  }
}
```

```
Mot de passe ? avoine
Mot de passe incorrect

Mot de passe ? Sesame
Mot de passe correct
```

Motivation (4)

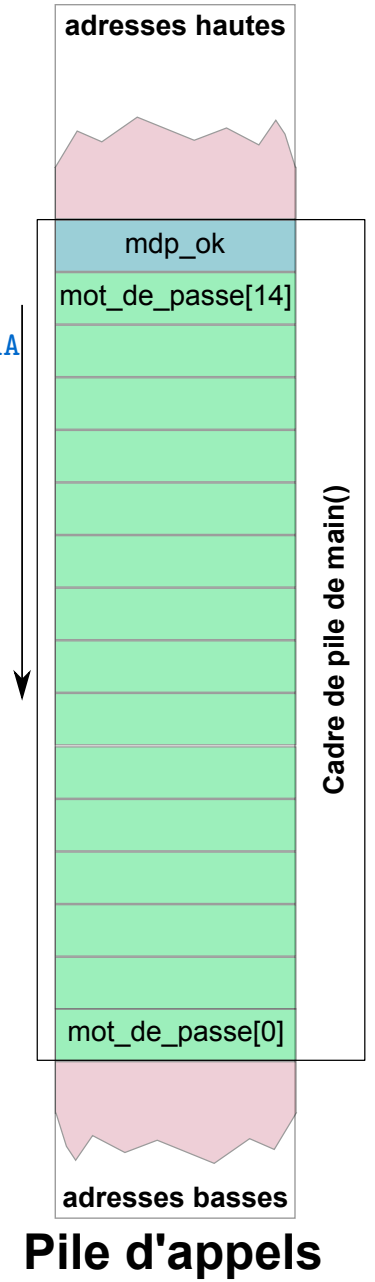
```
#include <iostream>
#include <cstring>

using namespace std;

int main(void)
{
  struct {
         char mot_de_passe[15];
         bool mdp_ok = false;
  } data;

  cout << "Mot de passe ? ";
  cin >> data.mot_de_passe;


  if (!strcmp(data.mot_de_passe, "Sesame")) {
    data.mdp_ok = true;
  }

  // [...]

  if (data.mdp_ok) {
    cout << "Mot de passe correct" << endl;
  } else {
    cout << "Mot de passe incorrect" << endl;
  }
}
```

```
Mot de passe ? avoine
Mot de passe incorrect

Mot de passe ? Sesame
Mot de passe correct

Mot de passe ? AAAAAAAAAAAAAAA
Mot de passe correct
```

Motivation (4)

```
#include <iostream>
#include <cstring>

using namespace std;

int main(void)
{
  struct {
         char mot_de_passe[15];
         bool mdp_ok = false;
  } data;

  cout << "Mot de passe ? ";
  cin >> data.mot_de_passe;


  if (!strcmp(data.mot_de_passe, "Sesame")) {
    data.mdp_ok = true;
  }

  // [...]

  if (data.mdp_ok) {
    cout << "Mot de passe correct" << endl;
  } else {
    cout << "Mot de passe incorrect" << endl;
  }
}
```

```
Mot de passe ? avoine
Mot de passe incorrect

Mot de passe ? Sesame
Mot de passe correct

Mot de passe ? AAAAAAAAAAAAAAAA
Mot de passe correct
```



Pile d'appels

Problème : Buffer overflow

Plan du cours

1. Représentation de l'information
2. Performances
3. Logique, circuits combinatoires et circuits séquentiels
4. Architecture du processeur MIPS
   (gestion des instructions, chemins de données)
5. Assembleur MIPS
6. Pipelines et caches