Langage C et Systèmes d'exploitation

Introduction

Responsable : Florian Brandner (4D59)
Auteur : Thomas Robert & Etienne Borde

Informations pratiques

- **Site Web de l'UE**
  - https://inf104.wp.imt.fr/
- **Points de contacts**
  - Le professeur de votre groupe
  - Le responsable de l'UE : Florian Brandner (4.D59)
    florian.brandner@telecom-paris.fr
- **Evaluation de l'UE:**
  - 1 Contrôle en fin d'UE

Quelques remarques sur les UEs INF104/INF106

- **Le but des deux UEs est de donner une introduction aux *Systèmes d'Exploitation***

  - Apprentissage ***rapide*** du Langage C (INF104 aka. LC), support au développement et à l'utilisation des services du SE. Donc peu de pratique du C dans un premier temps…

  - Apprentissage ***plus lent*** des principes/services du SE (INF106 aka. SE) l'occasion de pratiquer un peu plus le C et de mieux comprendre la première partie du cours.

  Donc… au début, ça va vite !!! Puis on ralentit le rythme donc accrochez vous et travaillez régulièrement: la deuxième partie du cours permettra de revenir sur ce qui est vu au début…

  Le jeu en vaut la chandelle: comprendre les mécanismes de base du fonctionnement des systèmes d'exploitation!

Rappel : un algorithme

- **Définition : description d'un ensemble d'opérations à réaliser sur des données**
- **Problème : les opérations utilisées dans la description de l'algorithme peuvent rester relativement abstraites**
- **=> Comment passe-t-on de cette description à sa réalisation ?**
- **Langage de programmation :**
  - une manière de référencer les données manipulées
  - leur représentation et structuration (e.g. tableaux, file …)
  - les opérations à appliquer aux données et leur impact

Un exemple d'algorithme avant de continuer…

- **Algorithme : calcul d'intersection de deux ensembles**
- **Types (mathématique) : ensemble (ENS)**
- **Inter(X,Y:ENS)**

{

$Z \leftarrow \emptyset$

pour chaque $x \in X$

{

$x \in Y \Rightarrow Z \leftarrow Z \cup \{x\}$

}

}

Affectation

Expression (logique)

Structure de contrôle d'exécution

? Et ça ?
Une sorte de
if then else ?

Savez vous répondre aux questions :

- Comment un ordinateur stocke les données ?
- Quelles opérations ***élémentaires*** sait faire un ordinateur ?
- Pourquoi passe-t-on par un langage de programmation ?
  *(pourquoi ne décrit on pas un programme en opérations **élémentaires**)*
- Quelles sont les étapes menant de la définition d'un programme à son exécution ?
- A quoi sert un système d'exploitation

Objectif : connaître les réponses et savoir les expliquer
(principe – limites/contraintes d'usage)

Quel est l'intérêt de connaître les réponses

- **Savoir ce qui faisable :** comprendre comment l'ordinateur fonctionne et va « exécuter vos programmes » permet de réaliser des applications plus complexes (en sachant ce qui est facile ou dur à réaliser)

- **Efficacité des traitements :** comprendre votre usage des ressources de calcul et stockage permet de faire des programmes plus efficaces

- **Se préparer à d'autres langages que C et JAVA** : ces concepts sont la base de l'informatique pratique, les connaître permet de se former plus facilement à de nouveaux langages

Suite de la séance

- **Introduction au concept de plateforme d'exécution**
  - Fonctionnalités et composants clés
  - Calculateur : principes de fonctionnement
- **De l'algorithme à son exécution**
  - Rappel des caractéristiques d'un langage de programmation
  - Stratégies d'exécution :
    compilation en instruction machine ou interprétation
- **Objectif: introduction et motivation des notions utilisées en apprentissage du C**
  - Représentation de l'information binaire
  - Adresse
  - Compilation

Support d'exécution une introduction

Fonctionnalités et exemples

- **Fonctions :**
  - Fournir les moyens de stocker de l'information
  - Fournir les moyens d'exécuter le traitement correspondant à un programme
  - Fournir des moyens d'interagir avec d'autres supports d'exécution ou l'environnement physique.

- **Exemples de plateformes différentes :**

Portables

Ordinateur de bord

Poste de travail

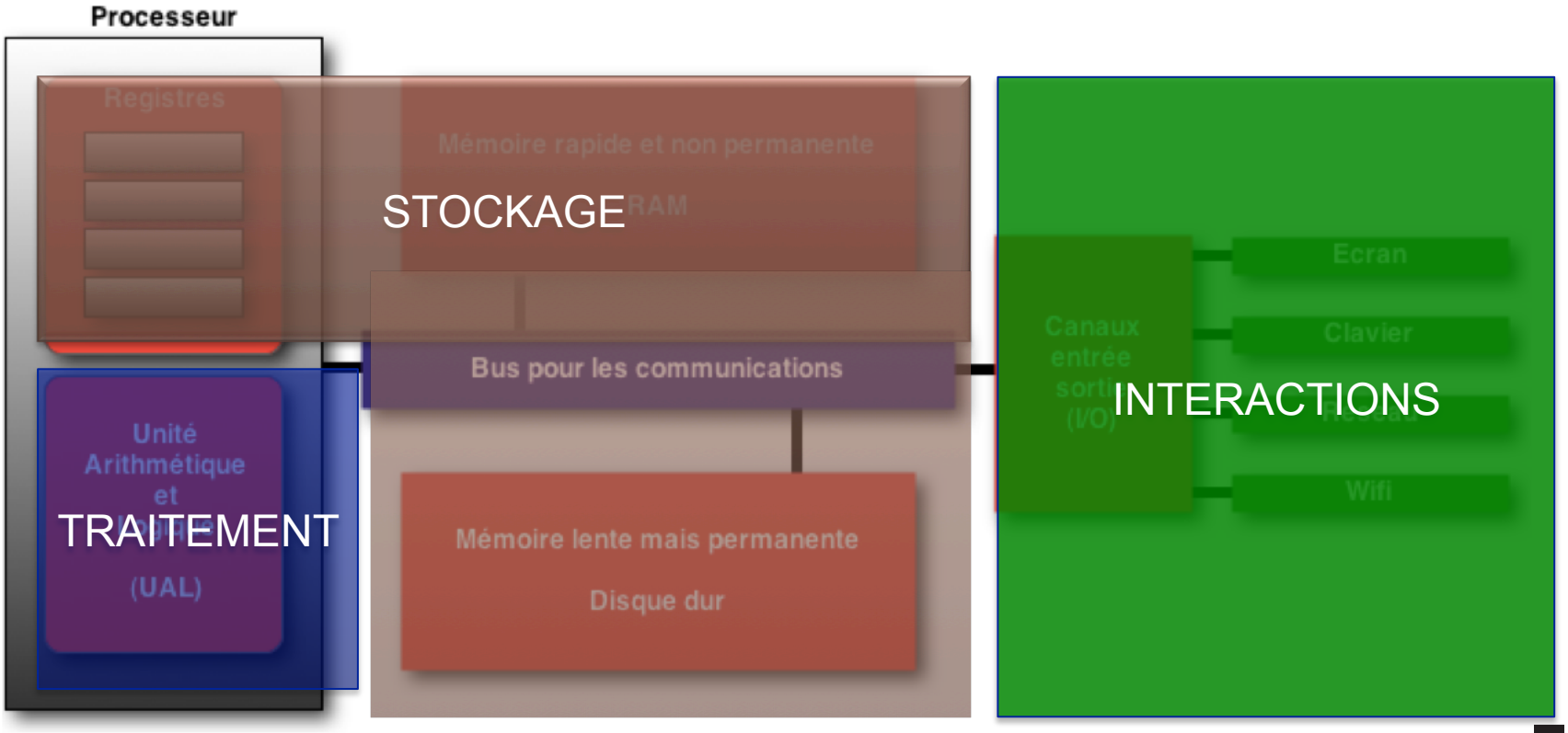
Un support d'exécution : les éléments clés

- **Un calculateur = regroupement de composants électroniques fournissant des capacités/ressources**
  - de stockage des données
  - de capture de stimuli extérieurs (physiques)
  - de communication de données avec d'autres calculateurs
  - de calcul et manipulation de données, contrôle d'exécution

- **Un système d'exploitation = logiciel pour organiser, simplifier et optimiser l'accès aux ressources**
  - Mise en place du partage et protection des ressources *(traitement / stockage)*
  - Simplification de l'usage et de l'accès aux ressources

Vision architecturale du calculateur

- **Intérêt : présenter une cartographie des ressources + de leurs interactions**



Stockage d'information

- **Un bit : unité de stockage d'information**
  peut prendre 2 valeurs : 0 et 1 (2 états possibles)
- **Les données manipulées par un processeur == séquences finies de bits**
  - 4 bits : 0110; 8 bits : 11010011…
- **Une séquence de N bits peut représenter n'importe quel élément d'un ensemble de taille inférieure à 2^N**
- **Ces séquences de bits peuvent être interprétées comme:**
  - Des nombres : 1001 = représentation base 2 de 9
  - Des mois : 000010000000 pour Mai

Une séquence de bits =
CE QUE VOUS DECIDEZ D'EN FAIRE (INTERPRETATION)

Un petit mot sur les systèmes numériques et le principe de représentation….

- **Séquence N bits interprétable comme un nombre en base 2**
- **ATTENTION : cela ne veut pas dire que l'on stocke toujours un nombre en mémoire**

  000010000000 voulait dire « Mai » et pas 2^7

⇒ **Indiquer à l'ordinateur comment interpréter les bits**

- **Représentations classiques :**
  - Système binaire : $b_7b_6….b_0$ ➔ $\sum_{i=0}^{7} b_i \cdot 2^i$
  - Complément à 2 : $b_7b_6….b_0$ ➔ $-(b_7 \cdot 2^7) + \left(\sum_{i=0}^{6} b_i \cdot 2^i \right)$
  - Symbole ASCII : des symboles « % » « A » « 7 » http://en.wikipedia.org/wiki/ASCII

Organisation du stockage

- **Organisation des bits en octet = 8 bits (granularité usuelle), plus pratique à écrire en base 16 (symboles: 0123456789ABCDEF)**

- **Problème du volume :**
  - Mozilla : 178 Mo de données utilisées dans certains cas
  - Base de données client : ~quelques Téra octets

Comment retrouver les données intéressantes dans un tel volume ?

- **Index et référence :**
  - Index : une fonction associant un identifiant à une donnée (mieux quand il est unique)
  - Référence : une valeur d'identifiant pouvant être utilisé pour récupérer la donnée qui lui est associée via l'index.
  - résoudre la référence = Index(Référence) →obtenir la valeur

Exemple Index – Référence

- **Identification des individus**
  - Index : Relation Identité – Numéro de sécurité sociale
  - Référence à une personne : 19303301213 77 (n° sécu)
- **Identification des données sur le « web » :**
  - Index : Relation URL – Donnée sur un serveur sur le web
  - Référence à une image : URL http://www.ici.fr/img.jpg
- **Identification d'un lieu (habitation)**
  - Index : Adresse Postale – Position géographique (lat,long)
  - Référence : 1 rue du lac, 75013 Paris

En pratique (1) mémoire

- **Mémoire (RAM) : composant permettant de stocker un ensemble d'octets de grande taille multiple d'une puissance de 2^K**

- **Caractéristiques :**
  - Index **:** 1 relation associant 1 **adresse** (i.e. un numéro) à chaque case mémoire de taille 1 octet
  - Référence à **un octet : adresse – contenu binaire de la case mémoire**

  - Remarques:
    - Granularité d'accès : groupe de 1 à 8 octets d'index consécutifs (ce groupe est appelé **mot mémoire**)
    - Référence à **plusieurs octets contigus :**
      1 adresse @ + nombre d'octets (1 à 8) à lire à partir de @

En pratique (2) Les autres stockages

- **Justification technologique de leur intérêt (et/ou):**

  Plus rapide, moins couteux utilisable directement par le processeur …
  beaucoup de raisons très variées

- **Registres : stockage mémorisant quelques octets**
  - taille = mot mémoire (8 octets sur une machine 64 bits)
  - Directement nommés => pas d'index à base de numéros
  - là où le processeur réalise ses calculs
- **Stockage de masse (disque dur) :**
  - Accès = bloc pour transferts vers ou depuis la mémoire (ensemble d'octet ~ 1000 octets)
  - Index structuré : (cas disque dur non SSD)
    - Plateau, cylindre … ~ adresse postale: rue, ville, pays

Bilan sur l'organisation du stockage

- **3 types de stockages essentiellement :**
  Registres, Mémoire, Stockage de masse (disque dur)

- **Notion d'index et de référence**
  - Index : le système d'association (id, valeur), l'id peut être défini en plusieurs morceaux
  - Référence : un id (permettant de retrouver une valeur si il est complet et à jour…)

- **Références aux données en mémoire par le processeur**
  - Index: adresse → 1 octet
  - Référence: adresse (i.e. un numéro)

Capacités de traitement

- **Composant clé : le processeur**
- **Traitements = instructions (opérations *élémentaires*)**
- **Jeu d'instructions = {des opérations élémentaires supportées par le processeur}**
  - Stockées en mémoire, donc
    - représentation binaire
    - Chaque instruction élémentaire à une adresse
  - Jeu d'instruction limitées
    - Calculs numériques simple (+ ; - ; * ; / …)
    - Transferts de données entre supports de stockage (mémoire → registre ou registre → registre ou registre mémoire)
    - Evaluations de conditions booléennes
    - Utilisent principalement les registres (pour des questions de performance)
      ….
- **Problème: Comment indiquer au processeur les instructions à exécuter ?**

Le cas particulier du Déroutement avec restauration

On veut calculer

$$\sum_{j=121}^{242} j$$

Rappel Somme 1…N = N*(N-1)/2
=> calcul possible en O(1)

idée N dans Ra et résultat dans Rb
Rb doit contenir ((Ra-1) * Ra)/2

Calcul de N*(N-1)/2 :

@	Instruction
911	Rb ← Ra-1
912	Rb ← Ra*Rb
913	Rb ← Rb/2
914	….
915	….

Le cas particulier du Déroutement avec restauration

On veut calculer

$$\sum_{j=121}^{242} j$$

Rappel Somme 1…N = N*(N-1)/2
=> calcul possible en O(1)

idée N dans Ra et résultat dans Rb
Rb doit contenir ((Ra-1) * Ra)/2

Calcul de N*(N-1)/2 :

@	Instruction
1	Ra ← 120
2	Call 911
3	Rc ← Rb
4	Ra← 242
5	Call 911
6	Rb ← Rb -Rc

@	Instruction
911	Rb ← Ra-1
912	Rb ← Ra*Rb
913	Rb ← Rb/2
914	Ret
915	….

Bilan sur les capacités d'exécution

- Représentation en mémoire de séquence d'opérations dont l'exécution est automatisée
- Capacité à définir la prochaine instruction à exécuter (e.g. déroutement)
- Capacité à réutiliser une même séquence d'instructions dans différents « contextes » (i.e. appel de fonctions)

Rappel :
Un support d'exécution : les éléments clés

- **Un calculateur = regroupement de composants électroniques fournissant des capacités/ressources**
  - de stockage des données
  - de capture de stimuli extérieurs (physiques)
  - de communication de données avec d'autres calculateurs
  - de calcul et manipulation de données, contrôle d'exécution

Problème : coder directement en langage machine
== construire un château de sable grain par grain…

faisable mais très fastidieux et technique

D'où l'intérêt des langages de programmation

De l’algorithme à son exécution

Langage de programmation = une convention d'écriture sur …

- **la définition des données (types, identifiants, visibilité, valeur..) et la manière d'y faire référence**

  int i=0; ou Eleve E;

- **les opérations que l'on s'autorise à utiliser sans avoir à en redéfinir systématiquement le sens**

  w← 3; (par exemple); ou « if (x<2) {x++;} »

- **la manière dont on définit des séquences d'instructions réutilisables (fonctions, méthodes) et leurs modes d'accès**

  this.set_x(4); vs set(x,4);

- **La structuration du programme**

  Classe / package / fichiers / modules et espaces de noms…

Pb de l'exécution d'un programme

- **Les éléments clés qui ont été abstraits :**
  - La nature des traitements et **leurs références**
  - La représentation des données et **leurs références**

- **Hypothèse : programme accessible en mémoire**

PB n°1 : la nature du traitement et leurs référence ne correspond pas en instructions en langage machine

PB n°2 : la représentation des données et de leurs références ne correspondent pas à des adresses

Stratégie de mise en œuvre

- **Interprétation (1 étape):**
  1. un ensemble d'instructions déjà en mémoire (le shell)
  - lit le programme (texte sous forme de séquence de commandes),
  - analyse les références (chemin vers un fichier),
  - réalise les traitements pas à pas

  Conséquence: analyse à refaire à chaque nouvelle exécution

- **Compilation et chargement pour exécution (2 étapes):**
  1. Compilation: traduit le programme (texte en C) en éléments équivalents du langage machine (adresses, instructions binaires)
  2. lancement de l'exécutable

  Conséquence: compilation à refaire à chaque modification du programme

  Un ensemble d'instructions (système d'exploitation) place la représentation binaire en mémoire pour en exécuter les instructions **(chargement + exécution du point d'entrée)**

Des exemples de langages et d'usages associés !!!

- **Programme à interpréter = script (ref. cinéma)**
  - Calculs et manipulation de textes : Lisp, perl
  - Langages web : PHP, scripts html
- **Langages compilés :**
  - C / C++ : programmation système / interface / robotique (beaucoup de programme en robotique en C++)
  - Ada : programmation systèmes critique (trains, avions, satellites…)
- **Ceux qui sont au milieu…**
  - Java : compilation des références et opérations complexes puis interprétation d'instruction génériques

La suite des cours INF104/INF106

- **Ce cours est dédiée au Langage C (LC), nous verrons**
  - Un introduction au langage C avec un point de vue « Comment on l'utilise »
  - Un approfondissement du langage C avec un point de vue « Comment ça fonctionne? »
  - Une introduction à la compilation, pour comprendre comment un LHN comme le C est traduit en langage machine
  - Un outil très pratique pour automatiser la production des programmes (make)

- **Un deuxième cours, dédiée aux Systèmes d'Exploitation (SE), nous réutiliserons le C pour utiliser les services fournis par un SE**

- **LC+SE = SELC, ne pas négliger l'importance de l'aspect SE (l'objectif principal des deux UE)**

Et avec tout cela on fait quoi

- **Presque tout ce que vous utilisez tous les jours**
  - Systèmes bureautiques
  - Tableau de contrôle d'un véhicule
  - Tableau de commande d'un robot outil
  - Serveur de transactions bancaires