

Université Paris 7 - Denis Diderot
Licence Sciences et Applications
Mention : Mathématiques, Informatique
et MASS
IF1 - Introduction à l'informatique et la
programmation
2006-2007

Jean-Marie Rifflet

Organisation générale :

- **Cours en amphi**
 - le jeudi 14h30-16h30, Amphi X2:
 - ◇ jeudi 28/09 et 12/10
(groupes "Informatique et Maths")
 - ◇ le jeudi 5/10 et 19/10
(groupes "Groupes passerelles" et "Mass")

Dates des cours suivants communiquées ultérieurement.

- **Cours-TD** : 2 heures par semaine
Début la semaine du 25 septembre
- **TP** :
 - 2h par semaine (TP1) : début la semaine du 25 septembre
 - 2h une semaine sur deux (TP2) : début soit le 25/09, soit semaine du 2/10

Page du cours :

www.pps.jussieu.fr/~rifflet/enseignements/IF1

Contrôle des connaissances :

Pa : Partiel le samedi 25/11/2005 matin

**Ej : Examen en janvier 2007
(absence éliminatoire)**

Er : Examen de rattrapage (juin/2007)

Td : Contrôle en TD (par défaut 0)

Tp : Contrôle en TP (par défaut 0)

Ecrit : $NE = \max (Ej, (Pa+Ej) / 2)$

Contrôle continu : $Cc = (Td + Tp) / 2$

Note finale session 1 : $(3NE + Cc) / 4$

Note finale session 2 :

$\max(Es, (3Es + Cc) / 4)$

Objectifs généraux de l'enseignement:

- **Connaissance élémentaire de l'organisation des ordinateurs**
- **Initiation à l'algorithmique et à la conception de programmes**
- **Apprentissage de la programmation impérative**
- **Initiation à l'approche objet**
- **Langage utilisé dans cet enseignement:
JAVA**

Références

- **Page Internet du cours :**

<http://www.pps.jussieu.fr/~rifflet/enseignements/IF1>

- informations générales relatives au cours
- copie des transparents

- **Concepts fondamentaux de l'informatique,**

A. Aho, J. Ullman,
Dunod, 1998. ISBN 2-10-003127-9

- **The Java Tutorial, third edition,**

Mary Campione, Kathy Walrath,
& Alison Hulm

Addison-Wesley, décembre 2000
Disponible sur Internet à l'URL

java.sun.com/docs/books/tutorial/books/3e/index.html

- **Thinking in Java,**

Bruce Eckel
Disponible sur Internet à l'URL

<http://www.mindview.net>

Internet et le Web

Internet : réseau (logique) d'interconnexion de réseaux physiques hétérogènes (réalisant la connexion physique de matériels)

- définition d'un ensemble de **protocoles** permettant la communication entre des réseaux (et des machines) hétérogènes
- mécanismes de désignation (adresses Internet) de bas niveau (sur 4 octets = 32 bits dans IPv4),

par exemple [134.157.168.9](#)

Protocole **IP** permettant également le routage de datagrammes d'un point à un autre

- protocoles de plus haut niveau permettant les échanges d'informations (**UDP**, **TCP**) entre applications ou entre utilisateurs (messagerie électronique, transfert de fichiers, . . .)
- autres protocoles tels que **DNS** pour la désignation symbolique des réseaux ou des machines (notion de **domaine**)
par exemple `fluor.pps.jussieu.fr`

Web, WWW, W3

définition d'un système sur le modèle client/serveur permettant l'échange d'informations hypertextes réparties sur des sites Internet (CERN 90)

Protocole **HTTP** (*HyperText Transfer protocol*)

Langage **HTML** (*HyperText Markup Language*): langage à balises permettant la mise en forme de documents hypertextes et rendant compte de la structure non linéaire des documents (notion d'ancrage et de lien hypertexte)

Types **MIME**
(*MultiPurpose Internet Mail Extension*)

utilisés pour définir le format du contenu des ressources accessibles (texte, image, audio, video, application)

Concept d'URL** (*Uniform resource Locator*)** : chaîne de caractères permettant de représenter une ressource sous un format universel et constitué de 5 parties :

- un nom de protocole (HTTP, FTP, TELNET, MAILTO, ...)
- un identifiant et mot de passe pour accéder à des serveurs sécurisés (déconseillé)
- nom du serveur: nom DNS du serveur hébergeant la ressource
- un numéro de port pour accéder au serveur (normalement 80 qui est omis)
- le chemin d'accès à la ressource sur le serveur

soit typiquement :

http://id:password@www.truc.machin:port/chemin

L'informatique

Pour l'Académie Française:

c'est la science du traitement rationnel, notamment par machines automatiques, de l'information considérée comme le support des connaissances et des communications, dans les domaines technique, économique et social

Il s'agit donc de traiter de l'information au moyen de machines (typiquement un ordinateur)

Quelques dates¹

- **1642** : la machine à additionner de Pascal
- **1812** : Babbage définit une architecture de machine proche des machines électroniques actuelles
- **années 1880** : machines à multiplier imprimant des résultats (Hollerith et Bull)
- **1944** : l'ENIAC, premier prototype de calculateur électronique (5000 additions par seconde, une multiplication en 3 millisecondes, quelque chose comme 18000 tubes électroniques, consommant autant que plusieurs rames de métro, 30 tonnes sur 1000 m^2 , fréquence d'horloge de 100kHz,
- **1947** : successeur de l'ENIAC, l'EDVAC (Mark I) intègre la totalité d'un programme entier (écrit en binaire sur un clavier) dans la mémoire de la machine (architecture de Von Neuman)

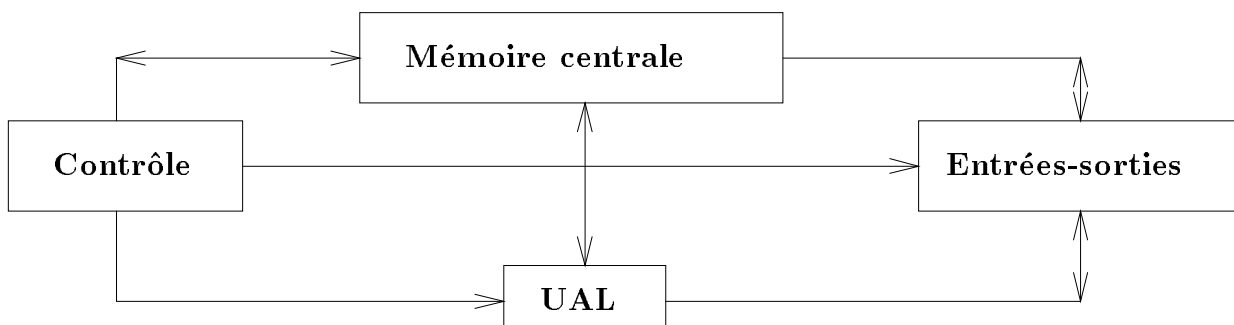
1. voir <http://www.grassouille.org/docs/cours-ii-html/node5.htm>

- **1950** : le Z4 fait des branchements conditionnels
- **1952** : l'EDVAC est achevé (1Mhz)
- **1958** : 1-er ordinateur à transistors [MIT] et premier circuit intégré chez *Texas Instrument*
- **1963** : invention de la souris (D. Engelbart)
- **1965** : Moore énonce sa loi:
les CPU doubleront de complexité tous les 18 mois
- **à partir de 1970** : apparition des microprocesseurs, des miniordinateurs et des microordinateurs
Génération successive de processeurs (Intel, Moorola et PPC) 8, 16, 32, 64 et 128 bits
Architecture RISC

Ordinateur de Von Neumann

Cinq composants:

1. unité arithmétique et logique (UAL) ou de calcul
2. unité de contrôle (ou de commande)
3. mémoire centrale ou principale (RAM)
4. unités d'entrée
5. unités de sortie



Processeur = union de 1 et 2

Unité centrale = union du processeur et de la mémoire

- **les jeux**
- **la bureautique (traitements de textes, tableurs)**
- **la communication (Internet)**
- **son, images et video**
- **les calculs mathématiques (météo, prévisions, balistique)**
- **bases et banques de données**
- **calculs massifs (machines parallèles et clusters)**
- **graphisme (CAO/PAO, reconnaissances de formes ou de codes)**
- **l'embarqué (automobile, avionique, téléphonie)**
- **la physique et le contrôle (simulation, résistance de matériaux)**
- **l'embarqué (automobile, avionique, téléphonie)**
- **la médecine (génomique, imagerie médicale, opérations)**
- **la robotique**

Les aspects logiciels

Machine nue a priori inutilisable ou très difficilement utilisable, d'où l'intérêt de disposer d'outils spécialisés

- **système d'exploitation** : il assure la gestion des ressources physiques (processeur, mémoire, périphériques) ou logiques (fichiers par exemple)
- **outils logiciels divers** : ils permettent l'utilisation plus ou moins conviviale, la réalisation de tâches spécifiques, le développement et la maintenance de nouvelles applications, . . . , en s'appuyant sur les services du système d'exploitation.

Par exemple: outils de bureautique (tableurs ou traitement de textes), compilateurs de langages, débogueurs, SGBD, . . .

Systeme d'exploitation

- Un système d'exploitation assure la gestion des ressources matérielles et logicielles sur une machine.
- principales évolutions:
 - ◇ traitement séquentiel des tâches (lecture d'un paquet de cartes perforées et exécution du programme lu)
 - ◇ multiprogrammation : lecture anticipée en mémoire (et donc cohabitation de plusieurs programmes en mémoire), toujours avec une exécution séquentielle
 - ◇ parallélisation des phases de calcul et d'entrées-sorties grâce au mécanisme d'interruption : les phases inertes du CPU pendant les E/S peuvent être récupérées pour une autre tâche

- ◇ **partage du temps** entre différentes tâches par ajout d'une horloge externe envoyant périodiquement une interruption
- ◇ **mémoire virtuelle** permettant d'utiliser plus de mémoire que n'en possède physiquement l'ordinateur (extension de la mémoire vive par une zone dite de **swap**)
- ◇ utilisation en mode **multi-utilisateurs**
- ◇ répartition des services (systèmes réseaux et systèmes répartis, clusters)
- **fonctions du système**
 - ◇ pilotes de périphériques
 - ◇ ordonnancement des tâches
 - ◇ gestion de la mémoire
 - ◇ gestion de fichiers

- **systèmes propriétaires/ systèmes libres**
 - ◇ **systèmes propriétaires des constructeurs longtemps prédominants**
 - ◇ **systèmes Windows de Microsoft sur PC**
 - ◇ **le phénomène Unix**
 - ce que c'est
 - la normalisation POSIX
 - les Unix propriétaires
 - les logiciels libres (Linux, FreeBSD)
- **systèmes temps-réel**
- **systèmes embarqués**

Interfaces utilisateur visant à permettre à un utilisateur de réaliser des actions (finalement exécuter des programmes)

- **aspects matériels :**

- ◇ clavier/écran/souris
- ◇ périphériques (disques, imprimantes, . . .)

- **aspects logiciels :**

- ◇ interfaces textuelles rudimentaires (shells Unix, interpréteur de commandes DOS)
- ◇ interfaces graphiques : interaction via la souris avec des boutons, des menus, . . .

Du point de vue d'un programmeur, possibilité de réutiliser des morceaux de code existant pour développer de nouvelles applications (bibliothèques, paquetages)

Langages de programmation

- ils permettent l'expression de programmes (l'équivalent des phrases pour une langue naturelle) compréhensibles par les ordinateurs, c'est-à-dire permettant l'expression d'algorithmes (suite d'opérations, d'actions, traitements, . . .) réalisables par un ordinateur
- les programmes écrits dans un langage doivent être non ambigus
- les langages définissent une syntaxe que doivent scrupuleusement respecter les programmes prétendument écrits dans ce langage

Niveaux de langages de programmation

- plus bas niveau (initialement pas d'autres) : le langage binaire (langage machine) permet l'écriture de programmes directement exécutables par l'ordinateur au travers de ses microprogrammes: programmation fastidieuse et programmes non portables

tout programme écrit dans un autre langage ne peut être exécuté directement: il nécessite soit sa traduction dans un programme écrit en langage machine, soit dans un langage intermédiaire interprétable par un programme particulier exécutable par l'ordinateur

- les langages d'assemblage (1950): définissent une notation symbolique. L'assemblage est l'opération de traduction correspondante
- les langages évolués: indépendants des machines, plus faciles à comprendre par l'humain. Les opérations de compilation et d'interprétation en permettent la compréhension par un ordinateur.

Différentes approches sont possibles :

- **compilation pure** :
un programme binaire directement exécutable est généré à partir d'un programme écrit dans le langage source
Un programme écrit en C est traduit par un compilateur, (par exemple le compilateur GNU gcc) en un binaire exécutable
- **compilation en langage intermédiaire** :
un programme d'un langage intermédiaire est construit qui sera exécutable (interprétable) par un programme particulier jouant le rôle d'une machine virtuelle.
Un programme écrit en JAVA est traduit par un compilateur Java (javac) en du bytecode exécutable par la JVM (Java Virtual Machine)

- **interprétation** :

un programme interprète à la volée les instructions d'un programme du langage source

Un fichier de commandes shell est interprété par un interpréteur du shell correspondant ou un programme Perl est interprété par un interpréteur Perl

Pour un langage donné, il peut exister sur un ordinateur à la fois un (ou plusieurs compilateurs) et un interpréteur.

Pour certains langages, le code (en langage intermédiaire par exemple en Java) peut être compilé lors de son exécution.

La jungle des langages évolués

- des disciplines de programmation, d'où des langages satisfaisant des slogans
 - ◇ décrire les structures de données
 - ◇ éviter les branchements (instruction goto)
 - ◇ éviter les effets de bord
 - ◇ manipuler les pointeurs avec précaution
 - ◇ écrire des assertions
 - ◇ commenter
- des styles de programmation : impérative, fonctionnelle, orientée objet, logique, formelle
- des types d'applications particuliers (calcul scientifique, logiciel de gestion, calcul formel, calcul logique, intelligence artificielle, logiciel de base, . . .)

Des styles de programmation

- la programmation impérative :

- ◇ langage machine de haut niveau pour une machine virtuelle basé sur l'effet de bord (opération appelée affectation) permettant de remplacer en mémoire une valeur par une autre.
- ◇ utilisation de noms symboliques pour les adresses
- ◇ un programme est une suite de traitements exécutés séquentiellement et possibilité
 - de réaliser des tests
 - itérer des traitements
 - modulariser le traitement (sous-programmes, procédures, fonctions)

◇ famille de langages illustrée (entre autres) par

- **FORTRAN** (*FORmula TRANslator*) [1957] pour les applications scientifiques (John Backus, IBM)
- **ALGOL** (*ALGOrithmic Language*) [1960]
- **COBOL** (*COmmon Business Oriented Language*) [1960] pour les applications de gestion
- **BASIC** (*Beginner's All-purpose Symbolic Instruction Code*) [1964] pour les ordinateurs personnels
- **PL1** (*Programming Language I*) [1964] (IBM)
- **PASCAL** [1968] (N. Wirth)
- **C** [1973] (Kernighan et Ritchie, Bell Labs)
- **ADA** [1983] (Ichbiah)
- **Perl** (*Practical Extraction and Report Language*) [1987]

- **la programmation fonctionnelle** :

- ◇ le principe y est de n'écrire que des fonctions et des appels de fonctions. On y privilégie la notion de calcul plutôt que celle de modification de l'état de la mémoire comme c'est le cas dans le style impératif
- ◇ fondé sur la théorie du λ -calcul (branche de la logique mathématique étudiant la notion générale de fonction indépendamment de ses domaines)
- ◇ notion d'environnement (ensemble de couples [nom,valeur]), dans lequel une paire ne peut pas être modifiée une fois créée

◇ **famille de langages illustrée de manière plus ou moins pure par**

- **LISP** (*LISt Processing*) [1959] (J. Mc Carthy, MIT) : typage statique (lors de l'exécution)
- **APL** (*A Programming Language*) [1962] (K. Iverson)
- **Scheme** [1975]
- **ML** (*MetaLanguage*) [1978] (R. Milner) et sa descendance (**Caml** [1984] et **Caml-light** [1990])
- **Miranda** [1985]
- **Haskell** [1990] (calcul retardé au maximum)

- **la programmation orientée objet** :

- ◇ **caractéristiques**

- **méthode de programmation et de conception de langage dont les buts sont de rendre les grands projets logiciels plus facile à gérer, améliorer la qualité et réduire le nombre d'échecs de projet.**
- **objet = unité de base d'un programme : il contient des données (attributs), et une série d'actions (méthodes) applicables aux données. D'où l'idée de modéliser un ensemble d'éléments d'une partie du monde réel en un ensemble d'entités informatiques (objets). De telles données informatiques regroupent les principales caractéristiques des éléments du monde réel (taille, la couleur, ...)**

◇ famille de langages illustrée par

- Simula [1967]
- SmallTalk [1972] (Xerox)
- C++ [1983] (B. Stroustrup)
- Eiffel [1985] (B. Meyer)
- Python [1990]
- Objective C [1983]
- Ocaml [1995] (*Objective caml*)
- JAVA [1995] (J. Gosling, Sun Microsystems).
- C#[2000] (C-sharp) (Microsoft)

- **la programmation logique** :
 - ◇ famille de langages illustrée par **PROLOG** (*PROgrammer en LOGique*) [1975] (Colmerauer) : adapté à l'IA (aux systèmes sur les langues naturels, systèmes experts)
- **les métalangages et langages de descriptions** : utilisés pour définir des langages et structurer des documents tels que
 - **SGML** *Standard Generalized Markup Language* [1986]
 - **XML** *Extensible Markup Language* [1998] (W3C)
 - **HTML** *Hyper Text Markup Language* [1992]
- **les langages de script** (les shells, Perl, Python, PHP, JavaScript)
- **les langages de requêtes** tels que **SQL** (*Structured Query Language*) pour communiquer avec une base de données
- **les langages de calcul formel** tels que **Mathematica** ou **Maple**

Des exemples de programme :

On souhaite calculer la somme des nombres entiers d'une suite donnée, c'est-à-dire $\sum_{i=1}^n t[i]$

- en Basic (avec goto) :

```

i=1
s=0
Loop:IF i=n GOTO END
    s=s+t[i]
    i=i+1
    GOTO Loop
END: PRINT s

```

- en APL:

```
+/t
```

- en MAPLE:

```
sum('t[i]', i=1..n) ;
```

- en C:

```

int somme(int t[ ], int n) {
    int i, s = 0;
    for(i=0; i<n; i++) s = s + t[i];
    return s;
}

```

- en JAVA:

```

class SommeListe {
    public static int sommeListe(int [] t, int n){
        int s = 0;
        for(int i=0; i<n; i++) s = s + t[i];
        return s; } }

```

- **en style fonctionnel (Caml):**

```
let rec sigma l =  
  match l with  
    [ ] -> 0  
  | h::t -> h + sigma t;;
```

ou encore:

```
let rec fold f r l =  
  match l with  
    [ ] -> r  
  | h::t -> fold f (f r h) t;;  
let sigma l =  
  fold (+) 0 l;;
```