

I – Les systèmes informatiques

1/ Quelques rudiments

1.1) Présentation

Avant d'aborder la programmation proprement dite, il est souhaitable d'avoir acquis quelques notions sur le fonctionnement d'un ordinateur.

Un ordinateur est une machine de traitement de l'information – on appelle *information* tout ensemble de données. Il est capable d'acquérir de l'information, de la stocker, de la transformer et de la restituer sous une autre forme. Il existe différents types d'ordinateurs (fixe, portable, serveur, station, ...) qui se distinguent par leurs domaines d'utilisation. Nous allons ici nous intéresser au fonctionnement d'un ordinateur fixe classique.

Remarque : le mot *informatique* vient de la contraction des mots *information* et *automatique*.

1.2) Le système d'exploitation

Le système d'exploitation d'un ordinateur est un ensemble de programmes qui dirigent l'utilisation des ressources matérielles de l'ordinateur (périphériques d'entrée / sortie, disques durs, etc ...) par les logiciels. Son rôle principal est de séparer les programmes et les détails matériels, ce qui simplifie grandement l'écriture des programmes d'application.

Plus précisément, le système d'exploitation a les responsabilités suivantes :

- donner l'illusion que l'ordinateur est multitâche (il stocke en mémoire les différentes applications que l'on veut utiliser et les exécute successivement par petits bouts) ;
- identifier les utilisateurs (chacun dispose d'un compte personnalisé) ;
- gérer la mémoire de masse (elle est généralement organisée en une structure arborescente de répertoires et de fichiers) ;
- contrôler l'accès aux différentes ressources de l'ordinateur ;
- gérer l'utilisation des différentes applications.

1.3) Utiliser un système de fichier

Les différents disques ont chacun leur propre système de répertoires et de fichiers. Le nom (ou chemin) complet d'un fichier s'obtient en enchaînant les noms des répertoires qui le contiennent, depuis le nom du disque (c'est une lettre majuscule), sous la forme contenant\contenu. Par exemple, le chemin complet de mon navigateur est

C:\Program Files (x86)\Mozilla Firefox\firefox.exe

On parle encore de **chemin absolu**. On peut aussi utiliser un **chemin relatif** (depuis le répertoire actuel) : le répertoire supérieur est alors noté `..`, comme dans `..\machin\truc`.

Il faut savoir manipuler l'arborescence des fichiers, soit à l'aide des fenêtres, soit au moyen d'une ligne de commande sous Windows ou d'une fenêtre *xterm* sous UNIX et les systèmes d'exploitation dérivés. Pour cela, on a besoin de quelques rudiments de DOS :

- `ls` : renvoie la liste des fichiers ainsi que le chemin complet du répertoire de travail.
- `pwd` : renvoie le chemin complet du répertoire de travail.
- `cd chemin` : change de répertoire de travail.
- `md chemin` : crée un répertoire.
- `rd chemin` : supprime un répertoire vide.
- `copy source destination` : effectue une copie d'un fichier.
- `del chemin` : supprime un fichier.

2/ Les langages informatiques

2.1) Généralités

Un *langage informatique*, contrairement aux *langages naturels* comme le français ou le breton, est un langage structuré utilisé pour décrire des actions exécutables par un ordinateur. Il est totalement dénué d'ambiguïté : tout le monde s'accorde sur ce que fait un programme donné (ce n'est pas toujours le cas avec les langages usuels).

Historiquement, le premier langage informatique a été **l'assembleur**. La programmation en assembleur est cependant fastidieuse, surtout pour des programmes importants. Plus grave, un programme écrit en assembleur dépend étroitement du type de machine pour lequel il a été écrit : si l'on désire l'adapter à une autre machine, il faut le réécrire entièrement.

Afin de répondre à ces problèmes, on a développé dès les années 1950 des langages de *plus haut niveau*. Dans ces langages, le programmeur écrit selon des règles strictes mais dispose d'instructions et de structures de données plus expressives qu'en assembleur, ce qui allège considérablement les codes des programmes.

2.2) Interpréteurs et compilateurs

On distingue grossièrement deux familles de langages informatiques.

Les langages interprétés

Un programme en langage interprété va être traduit en langage machine **au fur et à mesure de son exécution** par un *interpréteur*, c'est un programme chargé de décoder chaque instruction du langage et d'exécuter les actions correspondantes. L'avantage principal, c'est que le code est indépendant du système d'exploitation : il suffit d'installer l'interpréteur sur la machine et le code marchera. Par contre, c'est un peu lent.

Exemples : BASIC, LISP, Perl, Java, Python.

Les langages compilés

Dans le cas de langages compilés, la traduction en langage machine a lieu **une fois pour toutes** : un programme spécial, le *compilateur*, traduit chaque instruction du langage en une suite plus ou moins complexe d'instructions en langage machine. Les programmes compilés s'exécutent ainsi plus rapidement que les programmes interprétés, car la traduction est déjà faite et la machine se charge alors directement de leur exécution. On perd cependant en souplesse de programmation, car la compilation dépend du type de données, de l'architecture du processeur, du système d'exploitation, etc ...

Exemples : C/C++, Cobol, Fortran, Pascal.

2.3) Environnement de développement intégré

Un ordinateur sert ainsi à écrire et exécuter des programmes. Mais, pour faciliter la conception de programmes complexes, il est plus pratique de disposer dans un cadre unique de la plupart des outils dont on peut avoir besoin : ce cadre unique, c'est *l'environnement de développement intégré* ou IDE.

Pour simplifier, un IDE est un ensemble d'outils conçus pour fonctionner ensembles et destinés à la programmation dans un langage donné. Typiquement, un IDE contient :

- une *interface graphique* pour lancer les différents outils ;
- un *éditeur de texte* adapté au langage, qui permet d'écrire le code source ;
- un *compilateur*, qui permet d'exécuter les programmes ;
- un *débogueur*, qui permet de corriger les erreurs éventuelles

Il existe de nos jours de nombreux IDE (Visual Studio, Eclipse, ...) possédant chacun leurs spécificités. On peut noter que certains éditeurs de texte avancés comme Emacs ou Vim prennent en charge la coloration syntaxique, la compilation et le débogage et permettent ainsi de se dispenser d'un IDE.

Cette année, nous étudierons le langage Python et utiliserons les distributions Anaconda ou WinPython (uniquement sur PC). Elles sont fournies avec les différentes bibliothèques utiles et l’IDE Spyder, qui possède une interface à la fois complète et très simple à utiliser. La fenêtre de Spyder est divisée en trois parties :

- à gauche, *l’éditeur*, dans lequel on écrit les programmes que l’on souhaite exécuter ;
- en haut et à droite, *l’explorateur*, qui permet de se déplacer dans l’arborescence des fichiers, de contrôler le type et la valeur des variables utilisées, et qui peut aussi fournir de la documentation.
- en bas et à droite, *la console interactive*, dans laquelle vont s’exécuter les programmes (demande d’entrées, affichage des résultats ou des messages d’erreurs).

La console interactive permet d’utiliser Python comme une calculatrice : chaque ligne est immédiatement exécutée lorsque l’on tape Entrée. C’est idéal pour effectuer des opérations simples (calculs, affectations de variables, etc ...) ou utiliser des fonctions déjà en mémoire.

- Si l’on tape Ctrl + Entrée, on obtient une entrée sur plusieurs lignes.
L’exécution s’effectue alors en tapant Shift + Entrée.
- Pour interrompre une instruction qui bloque, il suffit de taper Ctrl + C.
On peut aussi cliquer sur le carré rouge dans la console.

Par contre, dès que l’on veut écrire un programme ou même une suite d’instructions dont on veut garder une trace, il faut impérativement utiliser l’éditeur.

La documentation est disponible à l’adresse <https://docs.python.org/fr/3.10/>