Systèmes informatiques

PTSI Lycée Eiffel

septembre 2017

1 Une brève histoire de l'informatique

Ce cours se concentrera assez rapidement sur l'étude de deux logiciels bien particuliers, mais avant de rentrer dans des détails plus techniques, nous allons tout de même consacrer quelques heures à un aperçu plus général de ce que représente l'informatique, et de la façon dont cela fonctionne. Vous vivez aujourd'hui, indiscutablement, dans un monde où l'informatique s'est à la fois révélée complètement indispensable et totalement intégrée dans notre vie de tous les jours. Qui peut imaginer en 2017 se passer d'un téléphone portable ou d'un accès à Internet ? Pourtant, ces inventions sont extrêmement récentes (votre cher professeur y avait à peine accès lors de ses années de lycée). Ce qui fait de l'informatique une science absolument fascinante, c'est que ses développements ont été fulgurants, alors même qu'elle n'est en fait qu'une suite assez logique de l'évolution continue de la capacité de l'humanité à communiquer et à transmettre des informations de façon efficace.

1.1 Un peu de vocabulaire

Rappelons en effet pour commencer ce que signifie le terme « informatique ». C'est un néologisme créé il y a environ 50 ans (la première utilisation du terme en France date de 1962) à partir des mots « information » et « automatique ». Il désigne donc au départ tout ce qui concerne le traitement automatique de l'information, c'est-à-dire à la fois toute la théorie du traitement de l'information, mais également son aspect pratique, et notamment le fonctionnement des ordinateurs. Le mot « ordinateur » a lui-même été inventé en 1955 par un professeur de lettres à la demande d'un responsable d'IBM. Les deux mots ont d'ailleurs failli faire l'objet d'un dépôt en tant que marques qui aurait empêché leur usage courant, mais ils ont échappé à ce triste sort (notons que ce ne fût par contre pas le cas du mot « informatics » en anglais, d'où l'utilisation outre-Atlantique du beaucoup moins général « computer science » pour désigner en gros tout ce qui est recouvert chez nous par le mot informatique). Pour autant, le traitement automatisé de l'information, ou du moins les premières recherches le concernant, ne datent pas de l'après-guerre.

1.2 Des premiers pas au premier ordinateur

- vers -3000 : invention du boulier.
- vers 830 : publication des oeuvres d'Al-Khwarizmi.
- 1642 : machine à calculer de Blaise Pascal.
- 1821 : machine à différences de Charles Babbage.
- 1936 : machine de Turing.
- fin des années 1940 : premiers « ordinateurs ».

Le problème principal de l'informatique est vieux comme le monde : la trasmission efficace de l'information. De ce point de vue, on peut donc considérer l'invention du **langage** (nous reviendrons sur ce concept un peu plus loin) comme la première pierre fondatrice de la science informatique. Viendra ensuite celle de l'écriture, qui permettra de communiquer des informations à distance, et de conserver l'information durablement. Depuis cette époque, comme nous le verrons un peu plus

loin, les grands principes n'ont guère changé, seuls le support et surtout l'efficacité du système ayant grandement évolué.

La création de systèmes automatisés de transmission de l'information, quant à elle, est issue de la convergence des progrès dans deux domaines distincts. Le premier est l'aspect théorique de la science qui n'était pas encore dénommée par le mot informatique, celle de l'automatisation de la transmission d'information. Les premiers prémisses sont apparus en mathématiques, avec la création d'**algorithmes** de calcul applicables de façon purement mécanique. Rappelons à ce sujet que le mot algorithme est dérivé de celui du mathématicien arabe Al-Khwarizmi, qui décrivit au neuvième siècle des méthodes de résolution d'équations du second degré (dans certains cas particuliers), même si l'algorithme tel que nous le concevons aujourd'hui est une création nettement plus récente (19ème siècle en gros). Le deuxième est tout simplement l'aspect technique. Impossible de construire une machine exécutant des algorithmes, même rudimentaires, sans appareillage, mécanique dans un premier temps, puis électronique dans un deuxième, complexe. Les premiers pas dans le domaine technique sont là encore apparus en mathématiques, comme aides au calcul. On peut remonter quelques millénaires en arrière pour constater l'utilisation du boulier (de façon apparemment indépendante) par plusieurs civilisations différentes, notamment en Chine. Les systèmes mécaniques se complexifieront et se diversifieront dans les siècles suivants, de la règle à calcul à la table de logarithmes (pas une machine à proprement parler). Un premier palier est franchi avec l'invention, en 1642, de la première calculatrice par un certain Blaise Pascal (oui, celui des *Pensées*, qui fut également un brillant mathématicien dont nous retrouverons le nom deux ou trois fois dans le cours de mathématiques cette année). Effectuant des additions et des soustractions à l'aide de rouages mécaniques, elle permettait aussi de simplifier les multiplications et les divisions, et sera suivies de plusieurs générations de machines améliorant sensiblement le concept sans le modifier profondément. Un siècle et demi plus tard, en 1801, Jacquard met au point son métier à tisser, commandé par des cartes perforées exécutant un programme, une notion qui rappellera peut-être de bons souvenirs à vos parents (grands-parents ?) s'ils ont connu l'avènement des premiers ordinateurs grand public. En combinant les inventions de Pascal et de Jacquard, le brittanique Charles Babbage conçoit une machine analytique qui préfigure l'ordinateur moderne, mais il ne dispose pas d'une technique suffisante pour en achever la construction.

Parallèlement, la théorie avance à grands pas. Outre la définition claire de ce qu'est un algorithme, le 19ème siècle voit l'apparition de la logique booléenne (logique à deux états, essentielle dans le développements d'outils fondés sur un système binaire). L'application de l'agèbre booléenne aux circuits électriques sera faite en 1938, dans la thèse de Shannon, qui invente accessoirement à cette occasion le terme bit (**bi**nary digi**t**). L'étape fondamentale suivante sera franchie par Alan Turing, le véritable père de l'informatique moderne. À l'origine mathématicien, il s'intéresse au problème de la calculabilité (comment savoir si une proposition donnée est démontrable ou non ?) et crée pour le résoudre son concept de machine de Turing, un modèle abstrait de fonctionnement d'un appareil mécanique de calcul. Tout est désormais en place pour l'avènement de l'ordinateur, dont l'évolution ne sera plus freinée que par les limitations techniques. Quelques inventions du début du vingtième siècle (notamment celle du tube à vide en 1904) permettent toutefois de créer les premiers calculateurs électro-mécaniques dans la première moitié du vingtième siècle. Leur développement est nettement accéléré par... la deuxième guerre mondiale, et la nécessité de disposer de machines puissantes (pour l'époque) pour décrypter les messages secrets ennemis. Nous verrons ailleurs dans ce cours que l'informatique doit beaucoup aux militaires, comme d'ailleurs plus généralement la science.

Le premier ordinateur entièrement électronique est généralement considéré comme étant l'ENIAC (**E**lectronic **N**umerical **I**ntegrator **A**nalyser and **C**omputer), construit à l'Université de Pennsylvanie entre 1944 et 1946, et financé par l'armée américaine. Ce mastodonte contenait pas moins de 17 468 tubes à vide, quelque 70 000 résistances et environ 5 millions de soudures (faites à la main !). Il pesait 30 tonnes pour une surface au sol de 167 mètres carrés. Ce n'est en fait pas le calculateur le plus performant de son époque, mais pour vous donner une idée, il permettait d'effectuer la multiplication de deux nombres à 10 chiffres en un millième de seconde (les meilleures machines électromécaniques descendaient difficilement en-dessous de la seconde, et à la main, eh bien, je vous laisse essayer !).

Pour la petite histoire, l'ENIAC n'utilisait pas un système binaire de représentation des nombres, mais décimal.

1.3 Les débuts de l'informatique grand public

- 1965 : loi de Moore.
- 1969 : invention du microprocesseur.
- 1969 : début de l'utilisation d'ARPANET, précurseur d'Internet.
- 1975 : commercialisation de l'Altair 8800.
- 1985 : première version de Windows.

Le développement des ordinateurs ne fera ensuite qu'aller en s'accélérant. L'invention du transistor en 1947, suivie de celle du circuit intégré en 1958, lancent les bases de la miniaturisation des ordinateurs. C'est en 1965 que Gordon Moore conjecture ce qui est connu depuis sous le nom de **lois de Moore** : une progression exponentielle de la complexité des semiconducteurs, et un doublement du nombre de transistors par puce tous les deux ans (énoncés très souvent repris de façon erronée sous la forme « la puissance doublée tous les 18 mois »). À ce jour, les lois de Moore continuent à être approximativement vérifiées. Moore est l'un des fondateurs (puis président) de la société Intel, qui crée en 1971 le premier **microprocesseur** (et reste aujourd'hui le numéro 1 mondial de la fabrication de processeurs). Les ordinateurs à taille humaine vont rapidement suivre.

Le premier ordinateur breveté en temps que micro-ordinateur est développé en France en 1973. Mais la première machine conçue pour être vendue à des particuliers est l'Altair 8800 en 1975. Son importance historique est primordiale, dans la mesure où parmi ses acquéreurs figurent Bill Gates, fondateur de Microsoft qui écrira pour cet ordinateur la première version du langage BASIC ; ainsi que le duo Steve Jobs et Steve Wozniak, qui créeront de leur côté Apple, dont les premiers micro-ordinateurs enverront très vite aux oubliettes de l'histoire l'Altair 8800 (ils seront notamment les premiers à deviner l'importance d'un périphérique balbutiant à la fin des années 70 : la **souris**). Microsoft, de son côté, se concentre sur la production de **logiciels**, puis celle de **systèmes d'exploitation**, développant d'abord MS-DOS puis Windows (premier système d'exploitation graphique) au début des années 80. L'ère de l'informatique grand public est réellement lancée.

1.4 L'explosion des technologies et de la commercialisation de l'informatique

- 1985 : premier ordinateur « portatif ».
- 1990 : première page web.
- 1992 : première version utilisable de Linux.
- 1994-1995 : premiers navigateurs web grand public.
- 1996 : première version de Windows CE, précurseur de Windows Mobile.
- 2007 : commercialisation du premier iPhone.

Plus le temps passe, plus la progression de l'informatique et rapide et... moins j'ai de commentaires à faire. Cette dernière période est celle dans laquelle nous nous trouvons encore : des ordinateurs de plus en plus puissants, de moins en moins chers, et de plus en plus nombreux. Parallèlement, les nouveautés les plus intéressantes de ces trois dernières décennies sont l'évolution du matériel (nous reviendrons un peu plus bas sur les périphériques et l'apparition des ordinateurs portables ; par ailleurs, vous êtes tous conscients du développement de nombreux outils technologiques qui ne sont plus des ordinateurs à proprement parler mais en partagent bien des caractéristiques, du smartphone à la tablette numérique en passant par la console de jeux), et la popularité grandissante du travail en réseau permis par Internet (là encore, plus de précisions sur ce sujet un peu plus loin). Aujourd'hui, un peu plus de 350 millions d'ordinateurs personnels sont vendus chaque année dans le monde.

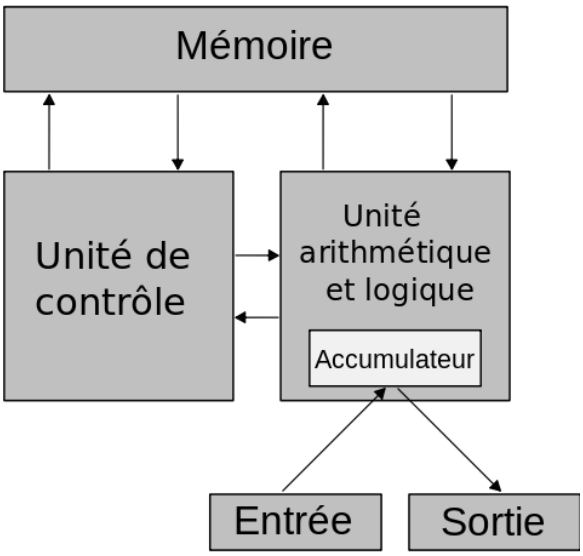
2 Fonctionnement d'un ordinateur

Toute transmission d'information suppose deux éléments indispensables : un moyen de stocker et de fixer l'information, et un moyen de la communiquer (de la transmettre). De ce point de vue, la seule chose qui a changé entre la transmission orale de légendes avant l'invention de l'écriture et l'envoi d'un mail en 2017 sont les deux éléments en question, le principe restant lui rigoureusement le même (on a évidemment énormément gagné en efficacité entre les deux). Profitons de cette remarque pour battre en brèche deux idées reçues sur l'informatique essentiellement dues à un manque de connaissances et à un vocabulaire peu rigoureux :

- le monde de l'informatique n'a strictement rien de virtuel. Certes, le stockage de l'information sur un disque dur est beaucoup plus économe en volume physique que par exemple le stockage sur papier qui a longtemps prévalu, mais toute information prend tout de même quelques micromètres carrés d'aluminium. La transmission elle-même n'est pas plus virtuelle que celle d'une formation transmise lors d'une conversation : si elle ne passe par un cable, elle fera intervenir des ondes certes invisibles mais qui ont une réalité physique indéniable. Un autre facteur pouvant expliquer cette sentation de virtualité est le caractère pratiquement infiniment recyclable de l'information stockée numériquement : on appuie sur une touche, puis sur une autre, et hop, sur le même espace mémoire, on a une information différente. Cette opération est évidemment nettement plus compliquée à réaliser lorsqu'on écrit un texte sur papier.
- la transmission des informations n'a rien d'instantané. Certes, on peut aujourd'hui avoir une discussion téléphonique d'un bout à l'autre de la planète en ayant l'impression d'instantanéité, mais en fait, la transmission est simplement devenue extrêmement rapide. Malgré tout, si vous tentez de télécharger (légalement, bien entendu) l'intégralité des films de Spielberg sur votre ordinateur, même votre oeil humain aura très largement le loisir de constater que la transmission prend un petit peu de temps. Le fantasme de la téléportation est encore loin.

2.1 Architecture générale

En plus de stocker de l'information, un ordinateur doit aussi être capable, rappelons-le, de calculer, c'est-à-dire, d'un point de vue plus rudimentaire, d'exécuter des programmes stockés dans sa mémoire. La structure générale de l'ordinateur a été décrite par l'architecture de Von Neumann peu avant la création des premiers prototypes comme l'ENIAC :



L'**unité arithmétique et logique** (UAL, ALU en anglais) est le coeur de la machine, qui effectue les calculs ; la **mémoire** contient à la fois les données et les programmes à effectuer par la machine, et l'**unité de contrôle** organise les programmes et ordonne les opérations à effectuer.

Quant aux **entrées-sorties**, comme vous l'auriez deviné, elles permettent de communiquer avec l'extérieur. De nos jours, ce modèle a peu évolué, à un détail près : dans l'architecture de von Neumann, les programmes sont stockés dans la mémoire sans faire de différence avec les données, ce qui permet notamment de modifier les instructions du programme en cours d'exécution. On préfère actuellement séparer plus nettement les données des programmes, et ne pas permettre cette modification dynamique des programmes. Actuellement, une unité centrale d'ordinateur est, sans rentrer dans les détails techniques, constituée des éléments suivants :

- la carte mère regroupe la plupart des composants essentiels au fonctionnement de l'ordinateur, notamment :
  - le microprocesseur, circuit intégré regroupant à la fois l'UAL et l'unité de contrôle au sens de von Neumann.
  - la mémoire, répartie entre mémoire vive (ou RAM pour **R**andom **A**ccess **M**emory) et ROM (**R**ead **O**nly **M**emory). La ROM stocke des informations qui resteront intactes même lorsque l'ordinateur n'est pas sous tension (tous vos dossiers de données sont dans la ROM) tandis que la RAM stocke temporairement des informations nécessaires au processeur pendant l'exécution des programmes. Pour cette raison, la RAM est d'accès beaucoup plus rapide que la ROM pour le processeur.
  - le chipset, qui gère les transmissions de données entre les différents composants.
  - les bus internes, qui connectent les différents composants.
- des bus externes reliant le microprocesseur à des connecteurs permettant eux-même le dialogue avec les périphériques d'entrée-sortie. Ces derniers servent à traduire une information complexe (mouvement de souris par exemple) en langage binaire compréhensible par la machine pour les périphériques d'entrée, ou au contraire traduire le langage binaire en programme exécutable (pour une imprimante par exemple) pour les périphériques de sortie. Les principaux connecteurs généralement disponibles sont :
  - les ports série et parallèle ont longtemps servi à connecter les différents périphériques, mais ont aujourd'hui laissé place au port USB (**U**niversal **S**erial **B**us) pour la connexion de la plupart des périphériques.
  - le connecteur RJ45 (RJ pour **R**egistered **J**ack) permet la connexion à un réseau (la plupart du temps Internet).
  - les connecteurs VGA, HDMI ou ATA permettent la connexion de périphériques spécifiques comme un écran, un téléviseur ou un disque dur externe.
  - les connecteurs d'extension permettent de brancher un élément améliorant généralement les capacités de l'ordinateur, comme une carte graphique.

  Notons que les connecteurs physiques ont de plus tendance à ne plus être utilisés sur les machines récentes, concurrencés par les systèmes de connexion sans fil Bluetooth ou Wifi. Le port USB sera certainement aussi inconnu de la prochaine génération qu'il n'a été une révolution formidable pour la mienne.
- les périphériques proprement dit (qui peuvent être intégrés à l'ordinateur dans le cas d'un portable, ou physiquement indépendants), parmi lesquels on peut citer :
  - pour les périphériques d'entrée, le clavier et la souris bien évidemment, mais aussi la webcam, le micro, ou le lecteur de DVD.
  - pour les périphériques de sortie, l'écran ou l'imprimante.
  - certains périphériques cumulent les fonctions, parmi les périphériques d'entrée-sortie on peut citer le disque dur externe, la clé USB ou même l'écran tactile.

2.2 Logiciels

Un **logiciel** (en anglais software, par opposition au hardware qui désigne le matériel informatique) est une liste d'instructions (ou programme) donnant accès à l'utilisateur à une interface simplifiée pour effectuer un certain type de tâches sur la machine (on parle de logiciels applicatifs) ou simplifiant simplement la communication avec la machine (logiciels système). L'utilisation de logiciels est

absolument indispensable dans la mesure où l'utilisateur est bien entendu incapable de communiquer directement avec le microprocesseur dans le seul langage que celui-ci connaisse, le binaire (nous reviendrons un peu plus loin sur ces histories de langage). Elle est d'ailleurs tellement indispensable que la carte-mère comprend elle-même un logiciel, le BIOS (**B**asic **I**nput **O**utput **S**ystem), qui permet d'effectuer quelques opérations élémentaires lors de la mise sous tension de l'ordinateur (et par exemple d'accéder à votre ordinateur le jour où vous avez vraiment salement planté votre Windows). Dans le même ordre d'idée, le système d'exploitation que nous évoquerons au paragraphe suivant est un logiciel sans lequel un ordinateur serait à peu près aussi utile qu'une voiture sans volant.

Les logiciels applicatifs sont extrêmement nombreux et variés. Citons par exemple les logiciels de jeu, les logiciels de bureautique (traitements de textes, tableurs) ou les logiciels de traitement d'image. Les langages de programmation, dont nous parlerons nettement plus intensivement bientôt, sont également des logiciels, même s'ils peuvent eux-même servir à créer d'autres logiciels. Historiquement, la qualité de l'offre logicielle est rapidement devenue un facteur de succès ou d'échec lors de la commercialisation des ordinateurs personnels. Un ordinateur performant sans logiciels pour l'accompagner a autant de chances de percer sur le marché qu'une console de jeux surpuissante, mais qui ne dispose pas d'une offre de jeux de qualité conséquente (à savoir très faible, pour ceux qui n'auraient pas compris). Une grande partie de la position dominante de Microsoft sur le marché vient d'ailleurs de la compréhension rapide de ce phénomène : la firme de Bill Gates a réussi à imposer à une vaste majorité d'utilisateurs ses logiciels phares en créant le concept de licence et en imposant la vente du logiciel couplée à celle de la machine. Une nouvelle philosophie s'est développées depuis une trentaine d'années, celle du logiciel libre, fondée sur les notions de gratuité du logiciel (à l'heure actuelle, la vente de logiciels rapport énormément plus que celle ce matériel informatique) et de partage du code (c'est-à-dire du programme constituant le logiciel), permettant à tous les utilisateurs de modifier eux-même le logiciel. À peu près tous les types de logiciels sont disponibles en version libre (sauf les jeux) à l'heure actuelle, mais leur part de marché reste très minoritaire.

2.3 Système d'exploitation et gestionnaire de fichiers

Le système d'exploitation (**O**pération **S**ystem ou OS en anglais), tout comme le BIOS décrit au paragraphe précédent, est un logiciel servant d'intermédiaire entre la machine et l'utilisateur. Plus précisément, le système d'exploitation organise toutes les demandes effectuées par les logiciels applicatifs à l'ordinateur : exploitation de la mémoire pour stocker les informations, du processeur pour les calculs etc. Il est en général présenté de nos jours sous forme d'interface graphique cliquable, mais ça n'a pas toujours été le cas, les systèmes d'exploitation plus anciens (MS-DOS notamment) nécessitant de taper au clavier des commandes compréhensibles par la machine. Contrairement au BIOS, et bien qu'il soit souvent vendu avec la machine, le système d'exploitation n'est pas intégré à la carte-mère et il est indépendant de l'architecture matérielle de l'ordinateur. Autrement dit, un PC par exemple n'est pas fait pour tourner sous Windows, et on peut très bien installer un autre système d'exploitation dessus (voire en avoir deux simultanément). Notons au passage qu'en plus du système d'exploitation, les machines sont souvent vendues avec des logiciels d'application (la suite Microsoft Office contenant Word et Excel dans le cas de Windows, par exemple), qui n'ont eux-même rien à voir avec le système d'exploitation, même si un logiciel donné est adapté à un système d'exploitation bien spécifique. Il existe aujourd'hui plusieurs dizaines de systèmes d'exploitation, certains libres, d'autres non (on parle de logiciels propriétaires). Les plus connus sont Windows, Mac OS (pour les propriétaires) et Linux (pour le libre). Il existe également depuis quelques années des systèmes d'exploitation spécialement conçus pour être utilisés sur d'autres types d'appareils numériques que les ordinateurs, comme Android pour les smartphones.

Profitons de ce paragraphe pour dire également quelques mots sur le système de gestion des fichiers, qui est intégré dans le système d'exploitation. C'est l'outil qui permet de gérer l'organisation du stockage d'informations dans la mémoire de l'ordinateur. Dans la carte-mère, la mémoire est simplement une succession de cases numérotées pouvant contenir de l'information binaire. Au niveau du gestionnaire de fichiers, cela fonctionne plus ou moins comme une base de données reliant les

noms des fichiers, leur taille et leur emplacement dans la mémoire (nous ne rentrerons pas plus dans les détails). Au niveau de l'utilisateur, je suppose que vous êtes tous familiers avec la présentation usuelle sous forme d'arborescence : un dossier contenant lui-même d'autres dossiers, qui contiennent eux-même des fichiers ou encore d'autres dossiers etc. Je me permets simplement de vous donner quelques petits conseils à propos de votre gestion de l'organisation des fichiers :

- utilisez vraiment le gestionnaire de fichiers, en n'hésitant pas à créer plein de dossiers pour retrouver plus rapidement vos fichiers. Si vous avez l'habitude de laisser trainer des fichiers sur votre bureau, ou d'avoir en tout et pour tout trois dossiers contenant chacun un millier de fichiers, vous ne gérez pas bien !
- donnez des noms clairs à vos fichiers. Ce n'est pas ça qui prendra de la place en mémoire, vous pouvez appeler un fichier TDinformatiquen°1_structuresconditionnelles, c'est très bien.
- si vous avez plusieurs versions d'un même travail en cours, et notamment si vous travaillez à plusieurs, numérotez les versions, histoire de savoir où vous en êtes et de pouvoir facilement revenir en arrière si les nouveautés ajoutées se révèlent finalement contre-productives. Si vous avez l'habitude de toujours sauvegarder votre fichier sous le même nom en écrasant la version précédente, là encore, vous avez tout faux.

Autre chose à propos des fichiers si vous travaillez cette fois-ci à plusieurs sur une même machine : la protection des fichiers. A priori, le fichier contenant simplement de l'information lisible dans la mémoire de l'ordinateur, n'importe qui ayant accès à l'ordinateur a accès au fichier. En pratique, c'est heureusement plus compliqué que cela, on peut restreindre l'accès au fichier à certains utilisateurs ou groupes d'utilisateurs (d'où l'utilité dans le cas d'une utilisation commune de machine de créer des comptes distincts pour tous les utilisateurs potentiels). Plus précisément, on peut effectuer trois actions sur un fichier : lecture (on se contente de visualiser le contenu du fichier), écriture (on modifie le contenu du fichier), et exécution (dans le cas de fichiers dits exécutables, par exemple un fichier contenant un logiciel). On peut séparer les restrictions concernant ces trois opérations. Ainsi, pour prendre un exemple qui empiète sur notre prochain paragraphe, un fichier html (page web) situé sur un serveur est en général lisible par tout le monde sur le web, mais n'est modifiable que par l'auteur de la page web. Sous Windows, l'accès à la protection de fichiers se fait simplement dans l'onglet Sécurité des propriétés du fichier (clic droit sur le fichier pour aller vite).

2.4 Réseaux et Internet

- WWW : **W**orld **W**ide **W**eb
- HTTP(S) : **H**yper**T**ext **T**ransfer **P**rotocol (**S**ecured)
- HTML : **H**yper**T**ext **M**arkup **L**anguage
- URL : **U**niform **R**essource **L**ocator
- TCP-IP : **T**ransmission **C**ontrol **P**rotocol - **I**nternet **P**rotocol

Le principe d'un réseau informatique est fort simple : connecter entre elles un plus ou moins grand nombre de machines pour leur permettre de partager de l'information. Le réseau que vous connaissez le mieux est le plus vaste qui existe aujourd'hui, à savoir Internet, mais on peut aussi très bien créer des réseaux à beaucoup plus petite échelle, par exemple pour permettre à tous les ordinateurs de notre salle de TP de partager des documents. Un réseau comme Internet est constitué de millions de points d'entrée (ordinateurs personnels des utilisateurs, ou serveurs web stockant des pages web) et du cablage nécessaire pour les relier. Sans rentrer dans les détails techniques, cela crée un maillage extrêmement complexe (l'image du filet a donné son nom au web). L'acheminement des données y est donc beaucoup plus complexe qu'au sein d'une machine où le bus représente par exemple l'unique chemin pour aller de la mémoire vive au microprocesseur. Le transport des données sur Internet (on parle de routage) est effectué par paquets (une quantité fixe de données est transférée à chaque étape) et gérée par un ensemble de protocoles connu sous le nom de TCP/IP. Ne comptez pas sur moi pour vous les décrire en détail, le sujet mériterait à lui tout seul un cours complet de plusieurs mois.

Donnons quand même quelques détails sur un sujet délicat, la sécurité sur Internet. Lorsque vous vous connectez au réseau, vous ne mettez bien sûr pas en partage la totalité des informations disponibles sur votre machine, mais simplement certains éléments indispensables à votre identification (notamment l'adresse IP de votre ordinateur). Par contre, vous utilisez sûrement des applications en ligne qui nécessitent le stockage d'un certain nombre d'informations vous concernant (contenu de mails, photos personnes etc). Comme dans le cas d'un ordinateur isolé, ce stockage n'a rien de virtuel et s'effectue donc sur des ordinateurs appartenant en général à la société gérant le logiciel ou le site sur lequel vous manipulez ces données (je vous laisse imaginer la quantité de serveurs qui tournent en permanence pour maintenir Facebook à jour). Même si vous passez un contrat avec la société en question, c'est elle qui gère les droits d'accès au fichier et décide du fonctionnement de la sécurité sur son site. Il ne vous viendrait pas à l'idée de déposer des copies de vos photos de soirée dans un casier à votre nom en libre accès à la bibliothèque municipale ? C'est pourtant ce que font certains sur Facebook. Alors, au moins, fermez le casier à clé (même si ce n'est pas vous qui détenez la clé).

3 Représentation des nombres et des données en informatique

Vous l'aurez désormais compris, le stockage des données est une problématique essentielle en informatique. Cela n'a d'ailleurs rien de spécifique à ce domaine, vous disposez vous-même à l'intérieur de votre crane d'une « machine » (le cerveau) dont l'une des spécificités est de pouvoir stocker de façon efficace de grandes quantités d'information. Dans le cas du cerveau, la façon dont ces informations sont stockées tient encore du mystère, mais pour l'ordinateur, on sait très bien comment ça fonctionne : toutes les données à l'intérieur de votre ordinateur sont ramenées, d'une façon ou d'une autre, à des suites de 0 et de 1, le fameux **binaire**. Pourquoi ce choix ? Une première nécessité était de ramener tous les types de données possibles (textes, images, programmes) sous une forme commune, pour des raisons évidentes de simplification de la gestion de la mémoire. Ce langage de stockage commun se devait d'être simple et surtout adapté à l'architecture matérielle de l'ordinateur. Celle-ci étant essentiellement constituée, au niveau le plus élémentaire, de dispositifs pouvant prendre deux valeurs facilement identifiables (interrupteur ouvert ou fermé, champ magnétique orienté dans un sens ou dans l'autre), le binaire s'est naturellement imposé (même si, on l'a vu plus haut, ça n'a pas forcément été le premier choix historique).

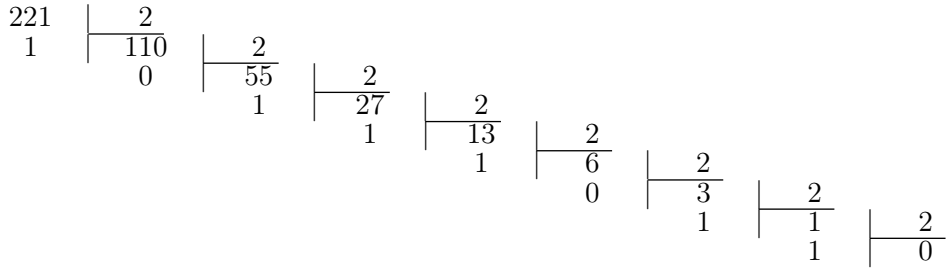
3.1 Bases et représentation des nombres entiers naturels

Ce choix du binaire a toutefois un inconvénient : il ne correspond pas à la convention (à peu près) universellement utilisée à l'heure actuelle pour l'écriture des nombres entiers, à savoir le système décimal. En pratique, comment un nombre est-il exactement enregistré dans la mémoire d'un ordinateur ? L'unité de base de mesure de la mémoire est le **bit**, qui correspond simplement à une unité d'information binaire, autrement dit un 0 ou un 1. Les bits sont ensuite regroupés en **octets**, qui sont une simple juxtaposition ordonnée de huit bits (les premiers étant traditionnellement appelés bits de poids fort, et les derniers bits de poids faible). Dans un octet, on peut stocker $2^8 = 256$ suites de 0 et de 1 différentes, donc 256 informations différentes. On peut donc y stocker naturellement, par exemple, les valeurs de tous les entiers naturels entre 0 et 255. La représentation binaire du nombre 221 sera ainsi :

1	1	0	1	1	1	0	1

(du moins si on utilise la représentation binaire usuelle, nous reviendrons sur ce point plus loin). Comment fait-on pour **convertir** un nombre sous écriture binaire en écriture décimale, et vice-versa ? Dans le sens binaire vers décimal, c'est assez simple : le chiffre le plus à droite (celui correspond au bit de poids le plus faible) est multiplié par 2^0, le suivant (de la droite vers la gauche) par 2^1, celui d'après par 2^2, et le premier chiffre par 2^7 (dans le codage d'un codage sur un octet). Ainsi, le nombre écrit plus haut vaut (en écriture décimale), $2^7+2^6+2^4+2^3+2^2+2^0 = 128+64+16+8+4+1 = 221$.

La conversion dans l'autre peut aussi être réalisée de façon assez élémentaire, en ayant recours à une technique que vous maîtrisez depuis le primaire : la division euclidienne. On divise le nombre à convertir par 2, on note le reste de la division (qui vaut nécessairement 0 ou 1), puis on recommence avec le quotient, jusqu'à ce que le quotient soit nul. Le nombre recherché a pour écriture binaire la suite des restes obtenus, en commençant par le dernier reste. Pour illustrer, écrivons ce que cela donne pour notre exemple précédent :



En lisant les restes de bas en haut, on retrouve 11011101. Ces techniques de conversion ne sont bien entendu pas spécifiques au passage de la base 10 à la base 2. Il suffit simplement de remplacer les puissances de 2 par des puissances de k (ou les divisions euclidiennes par 2 par des divisions par k), pour effectuer les conversions vers et depuis une base k quelconque.

Remarque 1. La conversion de base 10 en base 2 n'est pas la seule façon de coder les entiers de 0 à 255 sur un octet (ou plus généralement les entiers de 0 à $2^n - 1$ sur n bits). Il existe notamment un autre codage parfois utilisé en informatique, connu sous le nom de code **Gray** (ou code binaire réfléchi). Ce code est assez facile à reconstruire de manière récursive : 0 est toujours codé par le nombre binaire 0, 1 par la nombre binaire 1. Ensuite, on obtient facilement les nombres compris entre $2^n - 1$ et $2^{n+1} - 1$ en recopiant les $2^n - 1$ premiers nombres déjà obtenus et en ajoutant un 0 devant, puis en recopiant ces mêmes nombres en ordre inverse, en ajoutant un 1 devant. Autrement dit, en codage de Gray sur 2 bits, $0 = 00$, $1 = 01$ (on a simplement ajouté un 0 devant les deux nombres déjà codés), $2 = 11$ et $3 = 10$ (on recopie en sens inverse, 1 d'abord et 0 ensuite, et on rajoute un 1 devant). Sur trois bits, on obtient $0 = 000$, $1 = 001$, $2 = 011$, $3 = 010$, puis $4 = 110$, $5 = 111$, $6 = 101$ et $7 = 100$. Les plus curieux s'amuseront à vérifier qu'on peut obtenir directement le codage de Gray d'un entier décimal quelconque en effectuant le ou exclusif de son codage binaire (standard), avec ce même codage binaire décalé vers la droite (en ajoutant un 0 en premier bit). Quel est l'intérêt d'un tel codage ? Il a la propriété intéressante suivante : deux entiers consécutifs ont toujours un code de Gray très proche, plus précisément avec un seul bit de différence (c'est d'ailleurs également vrai pour le passage de $2^n - 1$ à 0).

3.2 Représentation des entiers relatifs.

Les codages présentés au paragraphe précédent ne permettaient de représenter que des nombres entiers naturels en machine. Pour représenter un entier relatif, on aura besoin de deux fois plus de place (en terme de capacité de stockage, ce qui représente en fait un seul bit supplémentaire), puisqu'il faut, en plus de la valeur absolue, préciser son signe. Une solution très simple consiste donc à sacrifier un bit pour indiquer le signe de notre entier, et coder la valeur absolue sur les bits restants. Ainsi, pour un codage sur un octet, on décide par exemple que le bit de poids fort codera le signe (1 pour un entier négatif, 0 pour un positif), et les sept autres bits la valeur absolue de notre entier. On pourra alors coder sur un octet tous les entiers relatifs compris entre -127 et 127, avec deux codages différents pour l'entier 0. C'est une solution qui n'est pas satisfaisante du tout d'un point de vue pratique, car l'addition binaire de nombres de signes opposés ne fonctionne plus du tout.

La solution la plus largement employée, et connue sous le nom de **complément à** 2, est subtilement différente : si on effectue le codage de nos entiers sur n bits, on convient que seuls les entiers naturels inférieurs ou égaux à $2^{n-1} - 1$ seront représentés par leur code binaire naturel (autrement dit, les entiers positifs ne sont codés que sur $n - 1$ bits, et ont tous un 0 en bit de poids fort).

Les autres codages servent à coder des entiers négatifs supérieurs ou égaux à -2^{n-1}, un tel entier $-a$ étant codé par le code binaire naturel de $2^n - a$. On constate avec cette méthode que le bit de poids fort indique tout de même le signe de l'entier, puisque tous les entiers négatifs auront un code commençant par un 1. Par exemple, notre codage binaire 11011101, qui représentait l'entier 221 tant qu'on se restreignait aux entiers naturels, sera désormais le code binaire de l'entier négatif $221 - 2^8 = -35$. Une autre façon de calculer les codes des entiers négatifs est de considérer le code de leur opposé, et d'en changer tous les bits sauf le dernier. Ainsi, 35 en binaire sur 8 bits se code 00100011, ce qui donne bien en changeant tous les bits sauf le dernier $-35 = 11011101$. Les plus courageux prouveront rigoureusement que cette façon de coder est compatible avec l'addition binaire. En fait, c'est très simple à voir si on considère que les entiers sont tout simplement codés sur n bits **modulo** 2^n, tout entier négatif ayant alors le même code que l'unique entier positif ayant même congruence que lui modulo 2^n. En tout cas, il faut bien garder à l'esprit que la taille maximale des entiers utilisable est imposée par le choix du nombre de bits sur lesquels seront codés ces mêmes entiers. Quelle que soit la puissance de l'ordinateur utilisé, $\mathbb{N}$ ou $\mathbb{Z}$ (et même $\mathbb{R}$, comme nous allons le voir) seront toujours pour l'informaticien des ensembles finis.

3.3 Représentation des nombres réels.

Les choses se compliquent encore nettement pour les nombres réels, où il faut coder la partie entière (pas de problème en utilisant ce qui précède), mais aussi ce qui se situe derrière la virgule (et indiquer d'une façon ou d'une autre où se situe cette virgule). Il est par ailleurs évident qu'on ne pourra représenter sur un nombre de bits donné qu'un nombre restreint de nombre réels, et donc imposer à l'avance une certaine précision. Sans trop rentrer dans les détails, décrivons rapidement la méthode généralement utilisée pour coder un nombre réel sur 32 bits (ce qu'on appelle du codage **simple précision** dans le jargon informatique, il existe aussi une version **double précision** codée sur 64 bits) : le bit de poids fort sert toujours à indiquer le signe du nombre, les 8 bits suivants, connus sous le nom d'**exposant**, codent un nombre entier relatif qui resprésente effectivement l'exposant d'une puissance de 2 par laquelle on va multiplier l'entier (naturel cette fois-ci) représenté par les 23 bits restants, qui portent le nom de **mantisse**. Ainsi, si notre premier bit est égal à 0, notre exposant code l'entier relatif -35, et notre mantisse l'entier naturel 2576453, l'ensemble des 32 bits codera le nombre réel $2576453 \times 2^{-35} \simeq 7.4985 \times 10^{-5}$. Cette méthode permet de coder des nombres dans un très grand intervalle (l'exposant peut théoriquement varier de -128 à 127, même si en pratique les deux valeurs extrêmes ne sont pas utilisées), mais avec une précision relative limitée : si on écrit notre nombre en notation binaire scientique (partie entière égale à 1, autrement dit décalage de la virgule juste derrière le premier chiffre significatif), seules les 23 premières décimales seront à prendre en compte. Attention bien entendu au fait que cette précision est relative : plus les nombres manipulés sont grands, plus les arrondis effectués vont eux-mêmes être de grands nombres.

3.4 Représentation en machine de caractères non numériques.

- ASCII : **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
- ISO : **I**nternational **O**rganization for **S**tandardisation
- UTF-8 : **U**nivers Character Set **T**ransformation **F**ormat - 8 bits

Savoir représenter des nombres est important, mais évidemment pas suffisant. Tous les autres types de données, on l'a déjà dit, devront être ramenés à un codage numérique, et plus précisément à un codage binaire. Pour terminer cette partie de cours, nous évoquerons simplement le codage des caractères autres que numériques, indispensable au stockage en mémoire de données sous forme de texte. Le principe en est toujours le même (fort simple) : associer à chaque caractère disponible un nombre binaire le représentant. Le nombre de caractères codables dépendra donc du nombre de codes disponibles, c'est-à-dire du nombre de bits réservés au codage d'un caractère. Ainsi, si on choisit de coder une liste de caractères sur 8 bits, on pourra représenter 256 caractères différents.

L'une des plus anciennes normes de codage de caractères (et d'une grande importance historique) est l'ASCII, qui date du début des années 60, et code sur 7 bits une petite centaine de caractères (certains codages sont réservés à des commandes de contrôle du type « saut de page ») correspondant à peu de choses près aux caractères disponibles sur une machine à écrire américaine classique (pas de caractères accentués notamment). De très nombreuses variantes de ce codage ont ensuite été créées, sur 7 ou 8 bits, permettant notamment d'ajouter les caractères accentués dans les pays les utilisant. Ainsi, en France, la norme de codage la plus utilisée pour le codage des caractères à l'heure actuelle est la norme ISO 8859-15 (parfois également appelée Latin 9 ou Latin 0, c'est une variante de la norme ISO 8859-1 alias Latin 1), qui permet l'affichage des caractères de l'alphabet latin, contenant également les caractères accentués apparaissant dans la plupart des langues européennes (langues nordiques notamment).

Ces normes de codage « locales » (il va de soi que le Latin 9 n'est d'aucune utilité pour un chinois) tendent toutefois à céder du terrain au profit d'une norme de codage internationale regroupant tous les caractères existants dans le monde : le **standard Unicode**. Les dernières versions de ce standard contiennent plus de 200 000 caractères (en particuliers tous les idéogrammes de tous les langues parlées en Chine, par exemple). Ce standard est associée à différentes normes de codage dont la plus répandue est l'utf-8, qui est en particulier compatible avec les autres normes de codage couramment utilisées (si vous configurez votre navigateur web préféré pour un affichage avec un encodage utf-8, il devrait afficher une page web correctement quelle que soit la langue utilisée). L'inconvénient de ce type de codage est évident : le très grand nombre de caractères disponible rend le stockage beaucoup plus lourd en termes de place mémoire utilisée.

Pour finir sur une note beacoup moins sérieuse, la norme de codage ASCII fait l'objet d'une nostalgie suffisante chez certains pour avoir motivé l'apparition d'une discipline « artistique » un peu spéciale : l'ASCII-art, dont le principe est de créer des images en utilisant uniquement les caractères de la table ASCII. Les smileys en mode texte du style :-) en sont des exemples très rudimentaires, mais certains malades ont été jusqu'à recréer des films en images ASCII-art animées (Star Wars en ASCII, le rêve absolu du geek qui sommeille en vous. Non ?).

4 Langages de programmation

4.1 Différents types de langages

Commençons par une définition très générale : le langage est (je cite Wikipedia !) « la capacité d'exprimer une pensée et de communiquer au moyen d'un système de signes doté d'une sémantique, et le plus souvent d'une syntaxe ». Autrement dit, un langage est constitué d'un alphabet (pas forcément des symboles écrits, cf. le langage des signes par exemple), de règles autorisant certaines constructions à partir de ces symboles et en interdisant d'autres (la **syntaxe** du langage), et surtout d'une **sémantique** donnant un sens aux assemblages de signes ainsi constitués. Toute communication (entre êtres humains ou non) nécessite le choix d'un langage pour transmettre le message, et bien sûr la connaissance de part et d'autre de la syntaxe et de la sémantique du langage. La notion de langage n'a bien sûr rien de scécifique à l'informatique, mais on trouve au sein de celle-ci quantité de langages dont les utilités peuvent être très diverses. Nous nous contenterons ici de donner une petite liste de certaines catégories de langages (parfois de simples exemples) pour illustrer cette diversité :

- les **langages naturels** sont ceux qui sont parlés par les êtres humains, par opposition aux langages formels utilisés en informatique. Ce lui que vous maîtrisez le mieux est bien sûr le français. Notons que ces langages naturels autorisent en général la communication même si le message est trabsmis avec une syntaxe incorrecte, ce qui ne sera pas du tout le cas avec un langage formel. Ainsi, dire à un enfant « Va brosser dents » devrait suffire à faire passer un message à l'interlocuteur (même s'il est déconseillé de s'exprimer ainsi si on veut que ledit interlocuteur emploie ensuite une syntaxe correcte !).
- il existe en fait d'autres langages « naturels », utilisés par exemple par les animaux.

- un **langage informatique** est un langage utilisé lors de l'exploitation d'un système d'information. Il ne s'agit pas nécessairement d'un langage de programmation. Ainsi, nous étudierons plus tard dans l'année le langage SQL, qui est un **langage de requêtes** destiné à effectuer des recherches dans une base de données. Ce cours est tapé dans le langage LaTeX (et mis en page automatiquement par le logiciel sous-jacent) que vous aurez sûrement l'occasion de manipuler également cette année. Un autre langage que vous connaissez tous et qui n'est pas un langage de programmation : le HTML, qui sert à écrire des pages web (pour vous faire une idée de la syntaxe de ce langage, qui utilise énormément le concept de balises, affichez sous votre navigateur web préféré le code source d'une page web pas trop compliquée).
- le **langage machine** est le langage, extrêmement sommaire, que peut directement comprendre un ordinateur. il est uniquement constitué de 0 et de 1 directement interprétables par le processeur.
- les **langages de programmation**, qui servent à écrire des algorithmes et des programmes capables de les exécuter. À peu près tous les logiciels que vous utilisez quotidiennement ont été programmés dans un de ces langages (il en existe énormément, pour les plus connus, citons l'historique BASIC, le C et son cousin le C++, Java et son compère Javascript, et bien sûr Python). Ces langages n'étant pas lisibles directement par notre ordinateur (non, non, personne ne programme directement en langage machine), ils sont traduits après écriture par un **compilateur** ou un **interpréteur** qui signalera les éventuelles erreurs de syntaxe (mais pas les erreurs de programmation, si vous écrivez un programme syntaxiquement correct mais qui ne fait pas ce que vous vouliez, ce n'est pas la machine qui peut s'en rendre compte).

4.2 Le langage de programmation Python

Parmi les centaines de langages de programmation disponibles, il faut faire un choix lorsqu'on décide un beau jour d'apprendre à écrire ses propres programmes. Pour cette année, le choix a été fait pour vous (et c'est un très bon choix !), ce sera la langage Python, créé en 1990 par Guido van Rossum et qui fait partie des langages de programmation les plus utilisés à l'heure actuelle. Il est par exemple utilisé par des entreprises telles que Google (où van Rossum a travaillé quelques années), la NASA ou Industrial Light and Magic (si vous ne connaissez pas, il s'agit d'une petite boîte spécialisée dans les effets spéciaux, qui a notamment créé ceux de Star Wars, de Terminator ou de Pirates des caraïbes). Bref, il s'agit d'un « vrai » langage utilisé par des spécialistes (auparavant on apprenait aux élèves à programmer en Pascal, qui est un très bon langage d'apprentissage mais qui avait le léger inconvénient de ne plus être utilisé que par des profs), mais dont la syntaxe est suffisamment simple et lisible pour être un langage idéal pour des débutants comme vous.

Pour rentrer un peu plus dans le détail, sachez que Python est un langage de programmation de haut niveau, à typage dynamique fort, libre, multi-plateformes et multi-paradigmes (orienté objet), doté d'une gestion d'exceptions et d'un ramasse-miettes (garbage collector). Vous n'avez rien compris ? Alors expliquons ces différents termes :

- **de haut niveau** : au sein des langages de programmation, on distingue les langages de bas niveau, dont la syntaxe est proche du langage machine et qui permettent de programmer en donnant des instructions très sommaires à la machine (par exemple réserver quelques octets dans la mémoire pour y stocker la valeur d'une variable), de ceux de haut niveau, dont la syntaxe est plus « naturelle » (avec utilisation de nombreux mots anglais), et où beaucoup de tâches fastidieuses sont automatisées.
- **typage dynamique fort** : nous reviendrons sur cette notion après avoir discuté des variables Python dans le prochain chapitre. L'idée est que Python gère les questions de typage (à quelle catégorie appartient une variable donnée) de manière automatique, contrairement à beaucoup d'autres langages où c'est le programmeur qui impose le type d'une variable.
- **libre** : j'ai déjà évoqué ce terme dans ce cours. Python est un logiciel gratuit (empressez-vous de l'installer sur votre ordinateur personnel), et son code source laissé à libre disposition de tout le monde.

- **multi-plateformes** : le langage est utilisable aussi bien sur un Mac que sur un PC Windows (ou avec d'autres OS plus exotiques).
- **multi-paradigmes** : il existe plusieurs philosophies fondamentales de la programmation, qui consistent à mettre en oeuvre des solutions différentes pour résoudre les problèmes. Sans rentrer dans le détail, les deux principales sont la programmation **impérative** (la plus classique), et la programmation **orientée objet**, qui consiste à voir toutes les variables d'un programme comme des objets qui vont interagir entre eux (Java fonctionne sur ce principe). Le langage Python est capable de gérer ces deux façons de programmer, même si nous ferons peu de programmation orientée objet cette année.
- **gestion d'exceptions** : Python est capable d'interrompre l'exécution d'un programme en cas de problème imprévu.
- **ramasse-miettes** : outil gérant de façon automatique les allocations mémoire et le nettoyage de la mémoire lorsqu'une variable n'est plus utilisée. Pour ceux qui ont déjà programmé en C, cela soulage d'un travail extrêmement fastidieux.

Et une dernière anecdote pour la route : pourquoi ce langage s'appelle-t-il Python ? Parce que son créateur était un fan des Monty Python.