Éléments d'informatique

Yannis Delmas

Éléments d'informatique

Yannis Delmas

Date de publication 3 octobre 2014

Table des matières

Avant-propos

Ce document rassemble des notes pour le cours «Éléments d'informatique» délivré en 2014-2015 aux premières années des masters *Web éditorial*, *esDoc* et *Ingénierie des médias pour l'éducation* de l'Université de Poitiers. Celui-ci vise à donner un arrière-plan culturel sur le fonctionnement des ordinateurs et systèmes d'exploitation ainsi que sur les types de fichiers usuellement employés sur le web.

Yannis Delmas

Chapitre 1. Éléments de théorie de l'information

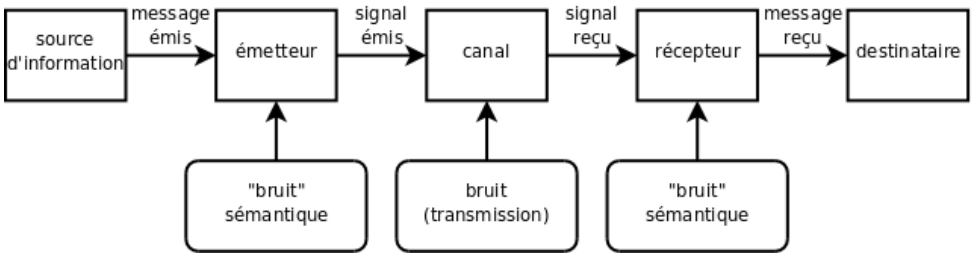
L'objet conceptuel central de l'informatique est l'information. De façon prépondérante cet objet est issu de la notion d' « architecture de von Neumann ». Il subit également l'influence, moindre aujourd'hui, de la « théorie de l'information » de Shannon. Cet objet prend également place dans le contexte plus large des théories du calcul, qui dépassent le champs de ce cours.

1. L'information selon Shannon

Définition simplifiée : *ordinateur* : machine universelle automatique du traitement de l'information, obéissant à des programmes constitués de suites d'opérations arithmétiques, logiques et de commande.

La théorie de Shannon est la première grande théorie systématique (et même mathématique) de la communication. Elle définit celle-ci comme un transfert d'*information* entre un *émetteur* et un *récepteur* à travers un *canal* de communication. Cette information ne peut être véhiculée que sous la forme d'un *code* commun aux deux parties. Les défauts de transmission de l'information par le canal sont appelés : *bruit*.

Figure 1.1. La théorie de l'information de Shannon et Weaver



Deux grands types de signaux : *analogique* : signal continu qui varie comme l'information qu'il représente (comme une analogie), *numérique* : signal discret (ou discontinu, ou nominal) qui code l'information sous forme symbolique.

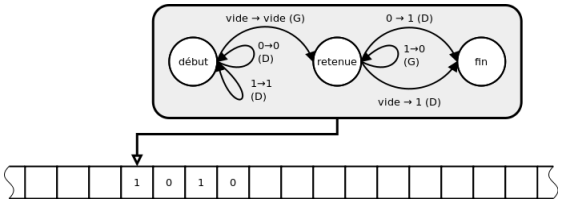
2. Le calcul selon Turing

On appelle *algorithme* la description méthodique, pas à pas et de façon explicite, d'un traitement ou d'une procédure.

Il existe de très nombreuses théories du calcul. Nous n'évoquerons que la plus simple à exposer : celle des *machines de Turing*. La machine de Turing est une formulation abstraite d'une manière universelle d'effectuer des calculs, c'est à dire de mettre en œuvre (on dit « *implémenter* ») des algorithmes. En ce sens, cela peut être une abstraction de l'action des ordinateurs concrets.

Une machine de Turing est, pour l'essentiel, composée de deux parties : un ruban permettant de stocker des *données* et un automate faisant office de *programme*. Le ruban est composé de cases successives susceptibles de contenir un nombre bien défini de symboles. L'automate, lui, fonctionne pas-à-pas. Il est composé d'un certain nombre de règles qui indiquent, pour chaque état de l'automate et chaque contenu d'une case du ruban, quelle case lire et quel état adopter pour le pas suivant.

Figure 1.2. Exemple de machine de Turing : +1 en binaire



Ceci est une première schématisation du travail d'un ordinateur : il traite de l'information sous forme numérique (les symboles sur le ruban) à l'aide de programmes (ici l'automate).

3. L'architecture de von Neumann

La plupart des modèles de calcul séparent de cette façon les programmes des données. Le principe de l'*architecture de von Neumann* est qu'il n'existe pas de séparation tranchée entre programmes et données. Le point fondamental est que ni les données ni les programmes ne sont des objets réels, pas plus que des idées. Ils ne sont que des *représentations codées* de ceux-ci. On parle donc plutôt d'information : l'information est un codage, d'objets, d'idées, de procédures, d'algorithmes, etc.

Exemple : un son est une suite de variations de pression de l'air. Représenter des sons sous forme d'information (pour en faire un fichier son) c'est une manière de coder ces variations. Dans le cas d'espèce on commence par échantillonner le niveau de pression au cours du temps. Ensuite, il s'agit de coder, pour chaque "tranche" de temps, l'amplitude de pression : on prend généralement la valeur de cette pression sur un intervale donné avec une résolution (nombre de valeurs possibles) donnée. Les nombres obtenus doivent être ensuite codés par des valeurs possibles de la mémoire. Chaque étape de codage est définie par des choix. Par exemple, on n'est pas obligé de représenter des amplitudes proches par des codes-mémoires proches : on choisit parfois des valeurs éloignées pour parer à certains défauts de transmission. Par ailleurs la méthode de codage des sons que nous venons de présenter est extrêment rudimentaire, les codages réels sont souvent beaucoup plus complexes, de façon à réduire la taille des fichiers grâce à la compression et à l'abandon d'information inutiles.

Le second point important à se représenter est que l'on peut passer sans solution de continuité de la notion de programme à celle de données. Un programme qui, dans un langage donné, sert à produire le texte « Bonjour tout le monde ! » n'est-il pas, en effet, une forme de codage de ce texte ? C'est d'ailleurs cette idée qui est à la base des principaux algorithmes de compression de données (sans perte) : un fichier compressé n'est finalement qu'une façon de programme permettant de produire les données non compressées. À l'inverse un document de nature essentiellement textuelle peut contenir des éléments programmés : "macros" pour les traitement texte, scripts intégrés pour une page *web* etc. Finalement, tout programme peut être vu comme une donnée et toute donnée comme une forme de programme (sur un mode très similaire à la dualité onde-corpuscule en Mécanique Quantique).

Dans les ordinateurs modernes l'information, quelle qu'elle soit, est donc stockée dans les mêmes *mémoires*, dont nous détaillons les types plus loin. Ces mémoires sont composées de cases, susceptibles d'accueillir un *contenu* et désignées par des adresses. Si l'on ne retient d'un ordinateur que "son" processeur et "sa" mémoire, on retrouve un schéma assez proche des machines de Turing théoriques.

4. Quantité d'information

Dans les premières mémoires vives (ce n'est plus toujours vrai aujourd'hui), l'information était représentée à l'aide de nombreuses bascules ayant chacune deux états. On les représente traditionnellement par « 0 ou 1 » mais il s'agit là d'un abus de langage, ou d'une vue de l'esprit. Les mémoires vives codent en fait un bit généralement par un état de passage ou non du courrant, les disques durs par un état d'aimantation etc., chaque support selon sa méthode. On pourrait donc aussi bien dire « blanc ou noir », « haut ou bas » etc.

La quantité permettant de désigner la place occupée en mémoire par une information, qu'il s'agisse de la *taille* du stockage ou de la *capacité* d'une mémoire, est appelée *quantité d'information*.

La plus petite quantité d'information concevable est donc un choix entre deux options. Elle peut se stocker dans une bascule. Pour cette raison, l'unité de quantité d'information est le *bit*, symbole : bit ou b, qui représente, précisément, le choix entre deux possibilités. Si l'on dispose de deux bits de mémoire (deux bascules), on peut coder quatre valeurs possibles : 00, 01, 10, 11. Avec trois bits, on code huit états possibles : 000, 001, 010, 011, 100, 101, 110, 111. Chaque bit supplémentaire multiplie les possibilités par deux. Le nombre d'états d'un systèmes à n bits est donc 2^n. Inversement, on peut définir la quantité d'information d'un système comme le logarithme à base 2 du nombre d'états possibles (si ce nombre n'est pas une puissance de 2, la quantité d'information ne sera pas entière).

Le multiple le plus courrant du bit est l'*octet*, en anglais *byte*, symbole o ou B. Il vaut huit bits.

Tableau 1.1. Quantité d'information et nombre de possibles

0 bit		2^0=1 (seul) possible	Il n'y a en fait pas réellement de choix
1 bit	un bit	2^1=2 possibles	
2 bit	deux bits	2^2=4 possibles	
3 bit	trois bits	2^3=8 possibles	
n bit		2^n possibles	
1 o = 8 bit	un octet	2^8=256 possibles	nombre de 0 à 255, ordre de grandeur : un caractère
2 o = 16 bit	deux octets	64 k = 65 536 possibles	
3 o = 24 bit	trois octets	16 M (env. 16 millions) possibles	ex.: une couleur (RVB)
4 o = 32 bit	quatre octets	4 G (env. 4 milliards) possibles	ex.: une couleur (RVBα, RVB32)
1 ko = 1024 o	un kilooctet	1024 = 2^{10}, mille octets, environ	ordre de grandeur : une page de texte
1 Mo = 1024 ko	un mégaoctet	1,05 millions d'octets, env.	ordre de grandeur : une disquette, une photo numérique, un roman, un morceau de musique en MP3
1 Go = 1024 Mo	un gigaoctet	1,07 milliards d'octets, env.	CD (0,8 Go), DVD (4,5 Go par face et par couche), mémoire flash (clé USB ou carte mémoire, 1 à 16 Go), ordre de grandeur : une encyclopédie, un film, un album musical en qualité CD
1 To = 1024 Go	un téraoctet	mille milliards d'octets, env.	disques durs usuels pour particuliers : 0,2 à 4 To
1 Po = 1024 To	un pétaoctet	1 Po = 2^{50}o	systèmes de stockage d'entreprise ; ajout quotidien à Facebook (2012) : 0,5 Po ; trai-

			tement quotidien de Google : 20 Po ; Internet archives (2010) : 6 Po
1 Eo = 1024 Po	un exaoctet	1 Eo = 2^{60}o	stockage de Facebook : 0,1 Eo ; données échangées par jour sur Internet (2009) : 0,5 Eo
1 Zo = 1024 Eo	un zettaoctet	1 Zo = 2^{70}o	données échangées par an sur Internet (2009) : 0,2 Zo
1 Yo = 1024 Zo	un yottaoctet	1 Yo = 2^{80}o	impossible à prédire (2^{80} représente en gros le nombre d'atomes dans quelques grammes de matière)

Attention : ces unités ne respectent pas le système international des unités (SI), pour qui les multiplicateurs sont des puissances de dix. Théoriquement, pour bien faire, il faudrait parler, pour les puissances de deux, de kibi-octet, mébi-octet etc. Les préfixes naturels sont : kibi (kilo binaire), mébi (méga binaire), gibi, tébi, pébi et exbi. La norme ne prévoit pas zébi et yobi, mais ils sont dans la lignée. Les préfixes symboliques sont : Ki, Mi, Gi, Ti, Pi, Ei, Zi et Yi. Pour tous ces préfixes, l'usage est fluctuant de mettre un tiret ou pas. La norme SI correcte n'est pratiquement utilisée que par certains (jeunes) informaticiens… et les vendeurs d'unités de stockage. Utiliser la norme SI présente pour ces derniers un double avantage : 1) ils vendent ainsi moins de capacité pour le même nom, 2) ils sont en conformité exacte avec la réglementation.

Chapitre 2. Le codage de l'information

1. Un exemple simple : coder les entiers positifs

Usuellement nous utilisons la base dix pour représenter les nombres, principalement parce que nous avons dix doigts… et du fait du poids de l'histoire. En revanche, dans le domaine informatique, il est parfois utile de travailler avec d'autres bases de numération.

Rappel : cent quarante cinq s'écrit « 145 » en base dix, ou décimale, parce que : $145=1\times10^2+4\times10+5$.

La première base de l'informatique est la base deux, ou binaire. Son intérêt est qu'écrire un nombre en binaire correspond à une série de 0 et de 1, donc naturellement à une écriture sous forme de série de bits. C'est d'ailleurs le codage habituel des entiers naturels.

Exemple : $81=64+16+1=2^6+2^4+1=1010001_b$ (on écrit parfois, en informatique, `0b1010001`). *Attention* : cette écriture ne se lit *pas* « dix millions dix mille un », mais « un zéro zéro un zéro zéro zéro un binaire ».

Pour s'entraîner au binaire, on pourra utiliser le petit jeu en ligne diffusé par Cisco : Cisco Binary Game.

Les informaticiens utilisent aussi parfois la base huit, ou octale… (mais c'est devenu rare aujourd'hui). Elle peut s'obtenir rapidement en groupant les chiffres binaires par trois.

Exemple : $81=1\times64+2\times8+1=1\times8^2+2\times8+1=121_o$ (on écrit parfois, en informatique, `0121`, *attention*).

Une autre base, la base seize ou hexadécimale, est particulièrement intéressante parce qu'elle permet d'écrire avec deux chiffres exactement les données codées sur un octet ($256=16\times16$). Comme $16=2^4$, chaque chiffre hexadécimal correspond à 4 chiffres binaires. Comme cette base comporte plus de dix chiffres, on complète les chiffres habituels par les premières lettres :

hexadécimal (base 16)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
binaire (base 2)	00000001	00100011	01000101	01100111	10001001	10101011	11001101	11101111								
décimal (base 10)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Exemple de conversion décimal→hexadécimal : $82=5\times16+2=52_h$. En programmation on écrit souvent : `0x52`.

Exemples de conversion héxadécimal→décimal : $A4_h=10\times16+4=164$, $33_h=3\times16+3=51$, $FF_h=15\times16+15=255$.

Si l'on représente un niveau, par exemple de couleur ou de transparence, comme dans les formats d'image habituels, par un octet, donc essentiellement un nombre entre 0 et 255, chaque niveau se représente par deux chiffres hexadécimaux. Voici quelques exemples :

hexadécimal	00	33	66	99	CC	FF
décimal	0	51	102	153	204	255
proportion	0 %	20 %	40 %	60 %	80 %	100 %

2. Coder les entiers relatifs

Nous avons vu que le codage des entiers naturels par une série de bits se fait simplement en prenant l'écriture binaire... quid des entiers négatifs ? La plus élémentaire, jamais utilisée en pratique, consiste à utiliser l'un des bits (le premier) pour représenter le signe. Si l'on prend l'exemple d'un octet, on a alors un bit de signe et sept bits pour coder la valeur absolue (un nombre indépendamment de son signe). Dans ce cas, un même octet peut représenter un nombre entre 0 et 255 ou un nombre entre -127 et +127 (exemple ci-dessous). Ce codage présente deux défauts : l'addition est complexe (il faut traiter quatre cas) et le zéro a deux représentations (-0 et +0).

Exemples :

- +82 codé sur un octet : « 0 1010010 »
- -82 codé sur un octet avec la méthode du bit de signe : « 1 1010010 »
- Les deux représentations de 0 sur 1 octet avec la méthode du bit de signe : « 0 0000000 » et « 1 0000000 »

Pour ces raisons la méthode du bit de signe seul n'est (presque) plus utilisée. Une méthode un peu plus efficiente consiste, pour coder -n, à prendre le complément à un du codage de n, c'est à dire de remplacer chaque 0 par un 1 et inversement. Si l'on prend toujours l'exemple du codage sur un octet, on aura toujours un nombre entre -127 et +127 et toujours deux représentations du zéro, mais l'addition est maintenant plus facile à calculer. On peut démontrer, en effet, que si l'on ajoute un nombre a et le complément à un d'un nombre b, on trouve a-b-1 (aux problèmes de dépassement près). Pour cette raison, la méthode (quasi) universellement choisie aujourd'hui pour représenter les entiers relatifs (positifs et négatifs) est celle du complément à deux : on représente un nombre négatif en prenant son complément à un, puis en ajoutant un à l'écriture binaire obtenue. Dans le cas d'un codage sur un octet, on peut ainsi représenter les nombre entiers de -128 à +127.

Exemples :

- +82 codé sur un octet : « *0*1010010 »
- complément à un de +82 sur 1 octet : « *1*0101101 » (même chose que 255-82)
- -82 codé sur 1o = complément à deux de +82 : « *1*0101110 » (même chose que 256-82)
- -82 codé sur deux octets : « *1*1111111 10101110 »

Si l'on travaille sur un octet, la représentation binaire de -82 (en tant qu'entier positif ou négatif) est donc la même que celle de 256-82 (en tant qu'entier positif), c'est à dire 174. Cela veut dire que « 10101110 » peut coder aussi bien le nombre -82 que le nombre 174. Il peut aussi coder de nombreuses autres choses, par exemple les caractères « ® » (codage latin-1 des caractères) ou « Ž » (codage latin-2 des caractères).

Les nombres à virgule et les grands entiers demandent, eux, des codages plus complexes.

3. Le codage des textes - première approche

Les caractères, comme tout le reste, doivent être transformés en séries de bits. Il y a fort longtemps… le codage des caractères se faisait uniquement sur 7 bit et seulement en caractères latins de base. Le code le plus courant sur les PC, et qui finit par s'imposer, fut le code ASCII, mais il y en avait de nombreux autres, concurrents. Autrement dit une même série de bits correspondait à des caractères différents pour des codages différents.

```
 Symb.  Déc.  Hex.   URI
espace=  32 - 0x20 - %20
    ! =  33 - 0x21 - %21
    " =  34 - 0x22 - %22
```

```
 # =  35 - 0x23 - %23
 $ =  36 - 0x24 - %24
 % =  37 - 0x25 - %25
 & =  38 - 0x26 - %26
 ' =  39 - 0x27 - %27
 ( =  40 - 0x28 - %28
 ) =  41 - 0x29 - %29
 * =  42 - 0x2A - %2A
 + =  43 - 0x2B - %2B
 , =  44 - 0x2C - %2C
 - =  45 - 0x2D - %2D
 . =  46 - 0x2E - %2E
 / =  47 - 0x2F - %2F
 0 =  48 - 0x30 - %30
 1 =  49 - 0x31 - %31
 2 =  50 - 0x32 - %32
...
 9 =  57 - 0x39 - %39
 : =  58 - 0x3A - %3A
 ; =  59 - 0x3B - %3B
 < =  60 - 0x3C - %3C
 = =  61 - 0x3D - %3D
 > =  62 - 0x3E - %3E
 ? =  63 - 0x3F - %3F
 @ =  64 - 0x40 - %40
 A =  65 - 0x41 - %41
 B =  66 - 0x42 - %42
 C =  67 - 0x43 - %43
...
 Z =  90 - 0x5A - %5A
 [ =  91 - 0x5B - %5B
 \ =  92 - 0x5C - %5C
 ] =  93 - 0x5D - %5D
 ^ =  94 - 0x5E - %5E
 _ =  95 - 0x5F - %5F
 ` =  96 - 0x60 - %60
 a =  97 - 0x61 - %61
 b =  98 - 0x62 - %62
 c =  99 - 0x63 - %63
...
 z = 122 - 0x7A - %7A
 { = 123 - 0x7B - %7B
 | = 124 - 0x7C - %7C
 } = 125 - 0x7D - %7D
 ~ = 126 - 0x7E - %7E
```

Avec l'extension de l'utilisation des ordinateurs sur toute la planète, et l'amélioration de la technique, on coda les caractères sur un octet, ce qui permet, théoriquement près de 256 caractères différents. Ceci ne permit pas encore d'avoir dans le même codage les caractères accentués Ouest-européens et les caractères grecs, par exemple. Pour écrire le Français, on utilisait souvent trois codages différents : le codage ISO-latin-1 (ou ISO-8859-1, standard), un codage proche spécifique à Windows (Windows-1252) et un autre spécifique à MacOS (MacRoman).

Ce principe n'était vraiment pas pratique : les documents ou mémoires de masse dans tel ou tel codage n'étaient pas forcément lisibles sur tel autre ordinateur. Certains caractères étaient tout simplement rendus incorrectement. Autre difficulté : quand l'euro a été introduit, il a fallu revoir les encodage... qui ne disposaient toujours pas tous du "o e dans l'o" (œ). La difficulté fut contournée en s'autorisant à coder les caractères sur plusieurs octets avec Unicode. Unicode, régulièrement révisé, norme ISO 10646, a vocation à comprendre tous les jeux de caractères existant ou ayant existé. Unicode correspond, en pratique, à plusieurs codages, dont le plus usuel est UTF-8. C'est aujourd'hui le code à utiliser ; les autres doivent être considérés comme obsolètes. Le code UTF-8 reprend le code ASCII pour les caractères ci-dessus ; les autres caractères étant codés sur plusieurs octets.

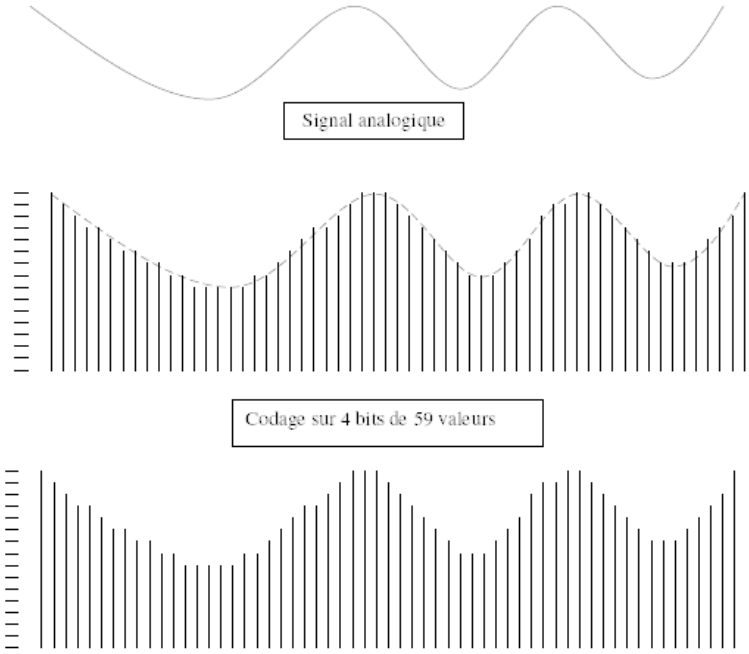
Nous verrons plus tard que les standards HTML et XML ont utilisé une autre approche du codage des caractères.

4. Échantillonnage : exemple des sons

On n'a évoqué ici que les informations numériques. Qu'en est-il quand on veut coder un signal analogique, un son par exemple (nous verrons d'autres exemples en détail dans la suite des enseignements) ?

La méthode la plus générale de "traduction" d'un signal analogique en signal numérique est l'échantillonnage. Voyons-en le principe sur l'exemple d'un son monophonique.

Figure 2.1. Exemple d'échantillonnage d'un son ([1])



Le signal est l'amplitude de ce son (reçue au moyen d'un microphone, par exemple) variant au cours du temps. Il y a deux problèmes pour une représentation numérique : l'amplitude et le temps sont deux grandeurs continue... qui demanderaient théoriquement une quantité d'information infinie. On fait donc un choix de représentation dégradée.

D'une part le temps est découpé en "tranches" d'égales durées, appelées *échantillons*. Cette durée, la *période d'échantillonnage*, correspond à une fréquence (son inverse) appelée *fréquence d'échantillonnage* (ou *résolution temporelle*). Par exemple, la "qualité CD" utilise un échantillonnage à 44,1 kHz, ce qui veut dire 44 100 échantillons par seconde. Les échantillons sont donc prélevés toutes les 22,7 µs environ. Les enregistrements professionnels vont jusqu'à 192 kHz.

D'autre part l'espace de variation du signal lui même, ici l'amplitude, est découpé en "niveaux", souvent équidistants. Ceci constitue une échelle discrète, numérique, appelée la *quantification*. Par exemple on peut coder sur quatre bits une amplitude sonore variant de -7 unités à +7 unités par pas de 1 unité. Plus la quantification est fine, meilleure est l'information sur le signal d'origine. Pour la "qualité CD", on quantifie l'amplitude sonore sur 16 bit, soit un peu plus de 65 mille niveaux.

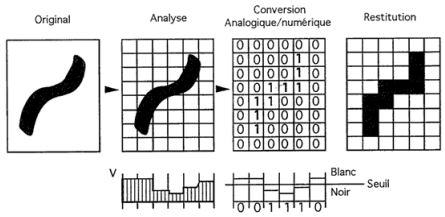
5. Échantillonnage multidimensionnel

Dans le cas des images fixes, le signal n'est plus une variation au cours du temps, mais une variation dans un plan. On ne découpe plus, cette fois des "tranches" de temps, mais des surfaces rectangulaires - souvent carrées. Les échantillons d'image sont appelés *pixel*s, symbole : px. Par abus on appelle aussi « pixel » l'unité de longueur correspondant à un côté de pixel.

[1] Image J.-P. Courjaud

Voici l'exemple d'une image quantifiée sur 1 bit :

Figure 2.2. Exemple d'échantillonnage d'une image ([2])



La finesse du découpage de l'espace est appelée *résolution* ou *définition*.

Si l'on parle d'un moniteur ou d'une image scannée ou destinée à l'impression, la résolution ou définition (plus exact) désigne le nombre de pixels par unité de longueur, généralement en « points par pouce », symbole : dpi (ou ppp, en français). Attention : pour une image de 3 cm × 5 cm à 90 dpi, ceci signifie : (3 cm×90 px/in÷2,54 in/cm) × (5 cm×90 px/in÷2,54 in/cm) = 106 × 177 = 18 762 pixels. Nombre de logiciels considèrent que la résolution des écrans est de 72 dpi. En réalité cette résolution est très variable, en fonction du matériel et des réglages choisis par l'utilisateur. Elle est plus souvent aux alentours de 90 ou 100 dpi pour les meilleurs moniteurs, avec les réglages les plus fins. Exemple : prenons un moniteur 4/3 de 15 in (38 cm) soit des dimensions de 12 in×9 in (théorème de Pythagore), en 800×600 la résolution est de 800/12 = 600/9 = 67 dpi et en 1024×768 elle est de 1024/12 = 768/9 = 85 dpi. Tout ceci suppose, bien entendu que les pixels soient carrés, sinon il faut parler de résolutions horizontale et verticale.

Si l'on parle d'une photographie ou d'une image purement immatérielle, cette notion de résolution n'a guère de sens. Le mot « résolution » désigne alors simplement les dimensions en pixels de l'image. Par exemple, la résolution d'une "photo numérique 4M pixels" est de 2 560 × 1 712 pixels. On écrira par exemple : 2 560 px × 1 712 px.

Si un signal a un domaine de variation à une dimension (1D), par exemple un son, sa quantité d'information varie comme la taille : deux fois plus de temps numérisé = deux fois plus de quantité d'information. En revanche, si le domaine de variation a deux dimensions (2D), comme une image, la quantité d'information varie comme le carré de la taille : pour une image deux fois plus grande, il faut compter quatre (2^2) fois plus de données. Pour les signaux 3D, la quantité d'information varie comme le cube de la taille.

6. Cas des signaux multidimensionnels

Le signal, lui aussi, peut avoir plusieurs dimensions. Par exemple un son peut être monophonique ou stéréophonique. Une image n'est généralement pas en noir-et-blanc (demi-teinte) mais peut prendre toutes les teintes perceptibles. Dans ce cas, pour la numérisation, on divise le signal multidimensionnel en *composantes* 1D. Pour un son stéréo, il s'agira des canaux "gauche" et "droite", par exemple. Pour une image en couleur, il s'agira des composantes "rouge", "vert", "bleu", par exemple. Les différentes composantes subissent ensuite la même quantification. Sur chaque échantillon, on a donc autant de valeurs numériques que de composantes.

Les images couleurs habituelles sont quantifiées à raison d'un octet par composante (256 niveaux), 24 bit en tout, soit 16 M teintes possibles pour chaque pixel. Les professionnels et certains scanner utilisent une quantification deux fois plus fine : 16 bit par composante (48 en tout), soit 64 k niveaux par composante.

La formule générale pour la quantité brute d'information est : $p = c.q.r = c \times q \times t^d$, où c est le nombre de composantes, q la quantité d'information nécessaire pour quantifier chaque échantillon, r la résolution, t la "taille", d le nombre de dimensions.

[2] Image IRHT CNRS, DR.

Ce que nous venons de décrire constitue le codage brut. Il est parfois très volumineux si la résolution est élevée. Souvent ce codage brut est suivi d'une compression, éventuellement avec perte, pour réduire la masse des données et éventuellement parer à des problèmes de transmission.

Exemple 1 : Une image couleur "4M pixels" au format "24×36" (c'est à dire un rapport 3/2) donnera une résolution de 2560×1712 pixels. Si l'on échantillonne à raison d'un octet par composante (rouge, vert, bleu) et par pixel, l'image brute (avant compression) pèsera 12,5 Mio, soit l'équivalent de 8 000 pages de texte, environ 40 romans. Avec le niveau de compression JPEG habituel des appareils photos, l'image enregistrée (avec pertes) ne pèsera plus qu'environ 1 Mio (dix fois moins).

Exemple 2 : Pour du son en qualité CD (44,1 kHz, 16 bit, stéréo), on a donc, chaque seconde 2×44 100 échantillons de 2 octets chacun, soit environ 176 ko/s. L'image de l'exemple 1 correspond donc à une minute de son environ (dans cette qualité). Avec un codage MP3 usuel, on comptera généralement une consommation 5 à 11 fois moindre : 16 à 32 ko/s, soit 128 à 256 kbit/s (on compte plus souvent en bits par seconde) pour un signal stéréophonique.

Chapitre 3. Types usuels de fichiers

On traitera ici les principaux types de fichiers usuels, à l'exclusion des fichiers multimédias, traités plus spécifiquement dans d'autres cours spécialisés.

Il est important de bien choisir le codage des données que l'on transmet à ses correspondants pour les raisons suivantes :

- Qualité/professionnalisme du service rendu : images lisibles, fichiers légers, mise en page conservée ou autre.
- Interopérabilité : les correspondant peuvent ouvrir les documents et pratiquer dessus les opérations attendues.
- Performance : aisance de travail, rapidité d'exécution, pas de perte de temps inutile etc.

1. Documents, formats et fichiers

Un fichier est généralement défini par deux éléments (au moins) : son nom et son contenu.

- Contenu.- Les informations stockées sont toujours codées à l'aide d'un type de fichiers particulier (GIF, PDF v. 2, PNG, XHTML 1.0, MS Word 2000 etc.). Les serveurs web détermine le plus souvent le type de fichier à l'aide de l'extension : c'est très rapide mais aussi assez approximatif (quelle version de HTML, de Word ?) voire, parfois, ambigu.
- Nom.- Il s'agit du nom proprement dit, suivi d'une ou plusieurs extensions (de un à quatre caractères, généralement) précédées chacune d'un point. Chaque extension donne des indications (pas toujours fiables) sur le format des données. Par exemple (fictif) : `abc.fr.html.gz` indique un fichier compressé contenant une page web rédigée principalement en français.

Traditionnellement, les extensions servaient à définir précisément le format de donnée des fichiers sur les ordinateurs de type PC-windows. Sur les ordinateurs de type MacIntosh, les fichiers contenaient en plus du contenu proprement dit un certain nombre de ressources, notamment des métadonnées, précisant : un icone[1], le format des données etc. Sur les ordinateurs Unix, cette information n'était généralement pas codée explicitement dans la mesure où chaque fichier commence généralement par quelques octets significatifs du format (appelés *magic numbers*). Par exemple les fichiers PDF en version 1.4 commencent par `%PDF-1.4`.

La diversité de ces procédés posait des problèmes pour l'interopérabilité entre les différents systèmes d'exploitation, provoquant des pertes de données ou des « scories ». Les difficultés les plus courantes concernaient l'ouverture des pièces jointes des courriers électroniques. De plus, aujourd'hui, les formats de fichiers sont trop nombreux pour que les méthodes traditionnelles soient suffisantes. On a donc défini une manière plus précise de spécifier le type d'encodage d'une donnée : le *type MIME*. Le registre MIME référence tous les principaux formats de données. Pour un fichier HTML, le type MIME est : `text/html`. De façon générale un type MIME est composé de deux parties : le type principal (`text` pour le cas précédent), suivi d'un sous-type (`html` dans ce cas). Ces deux informations sont éventuellement complétées d'informations complémentaires (jeu de caractère ou version du format, le plus souvent).

Les principaux types MIME :

- text : textes, au sens large (`text/plain`, `text/html`, `text/xml`, `text/rtf`,...),
- image : `image/jpeg`, `image/gif`, `image/png`,

[1] Le mot féminin « icône » est désormais réservé au domaine religieux. Pour l'informatique on utilise maintenant le mot masculin « icone ».

- audio : `audio/mpeg` (pour le MP3),
- video : `video/mpeg`,
- message : `message/rfc822` (courrier électronique),
- multipart : regroupement de parties de nature différente (pour des pièces jointes de courriers électroniques),
- application : données dans un format spécifique à une application donnée,
- model : description de scène 3D (`model/vrml`).

Les types MIME permettent de spécifier les données dans des contextes très variés. Initialement, il fut créé pour le courrier électronique : MIME signifie « Multipurpose Internet Mail Extensions », pour préciser le format du message et des pièces jointes. Ces types MIME peuvent être complétés d'autres indications : jeu de caractère, nom des pièces jointes etc.

Aujourd'hui, le type MIME est utilisé par les serveurs web pour indiquer aux clients web (les navigateurs) le format des données envoyées. Ces navigateurs appliquent scrupuleusement cette information, pour des raisons d'efficacité. Toujours pour des raisons d'efficacité les serveurs web déterminent le format des données le plus souvent d'après l'extension des noms de fichiers. Il est donc impératif de faire extrêmement attention à cette information. Les serveurs les plus évolués peuvent être configurés pour utiliser la méthode Unix, mais ceci est plus long et peut donc nuire au service rendu.

2. Archives et fichiers compressés

Les archives et systèmes de compressions répondent à plusieurs types de besoins :

- Archivage et sauvegarde : L'objectif est de grouper plusieurs fichier en un seul, avec ou sans compression. La qualité à rechercher est la pérennité, on favorisera donc les formats fiables, répandus, standards (de fait).
- Compression pour faciliter une communication : L'objectif est d'économiser de la bande passante ou du temps de téléchargement. La qualité à rechercher est l'interopérabilité, pour que tous les correspondants potentiels puissent décompresser le document dans de bonnes conditions.
- Compression pour archives personnelles : On pourra, dans ce cas, favoriser des formats compressant plus efficacement, éventuellement au prix d'un temps de compression important.

Quelques points à garder à l'esprit concernant l'archivage :

- La loi de Moore, c'est à dire la multiplication rapide de la capacité des mémoires de masse, rend souvent inutile la compression.
- Les CD et DVD usuels ne sont pas fiables au delà de dix ans. Il peut être utile de préférer une sauvegarde “vive” (sur disque dur) pour le plus long terme.
- Les mémoires flash ne sont pas adaptées (à ce jour) pour l'archivage (il faut les alimenter régulièrement, en les plaçant dans un appareil).
- Les données importantes doivent toujours être stockées en deux exemplaires au moins, sur deux supports différents.

2.1. Formats de compression

Principaux formats de compression pure :

- Z (logiciel Unix `compress`) - obsolète

- zip (algorithme “deflate“) - courant sous Windows, intégré aux principaux navigateurs web
- GZ (logiciel GnuZip) - courant sous Unix/Linux, intégré aux principaux navigateurs web
- BZ2 (logiciel `bzip2`) - surtout sous Unix/Linux, celui qui compresse le plus (au prix du temps de calcul de la compression), décompression rapide

Ces systèmes peuvent être employés pour compresser un fichier unique (par exemple une archive pure) ou une télécommunication (par exemple un échange entre serveur web et navigateurs).

2.2. Formats d'archivage

Il ne faut pas confondre archivage et compression. L'archivage consiste à grouper plusieurs fichier en un seul. Certains format permettent de compresser les données en même temps, d'autres pas. Dans ce second cas, on peut combiner l'archivage avec un système de compression pur.

Principaux formats d'archives :

- `ar` - obsolète, pour les anciennes bandes magnétiques
- `tar` - usuel sous Unix/Linux, ne compresse pas, souvent combiné avec une compression (`tgz` = `tar.gz`, `tz2` = `tar.bz2`)
- zip - usuel sous Windows, la plupart des systèmes le comprennent nativement, intègre la compression
- autres (à éviter, en général, pour des raisons d'interopérabilité et de pérennité) : ace, rar, 7-zip...

3. Documents textuels

3.1. PDF

Le format *PDF* (Portable Document Format, extension : `pdf`, type MIME : `application/pdf`) est une évolution du format PostScript (extension : `ps`, type MIME : `application/postscript`), tous deux créés par la société Adobe. PDF est maintenant un format ouvert normalisé par l'ISO (sous le nom PDF/A-1 ou ISO-19005-1), même si certaines extensions restent la propriété d'Adobe. On peut lire les documents PDF à l'aide de nombreux logiciels, dont le principal est Adobe Reader (anciennement Acrobat Reader).

Les formats PostScript et PDF sont essentiellement des formats de description de page. Ils décrivent chaque page d'un document en précisant où placer quoi. C'est un format très général, susceptible de contenir tous types de textes ou d'images fixes.

Le PDF est destiné à représenter des documents à imprimer (sans modifications) ou des formulaires à renseigner (généralement pour impression, éventuellement pour télétransmission). C'est, en particulier, *le* format privilégié de la GED (gestion électronique de document). Il est recommandé pour la GED par la DGME (Direction Générale de la Modernisation de l’État).

Attention : si on veut que les pages d'un fichier PDF soient rendues exactement comme chez leur concepteur, il est nécessaire que ce fichier inclue toutes les fontes (dessins des caractères) employés. Ceci est généralement l'option par défaut des principaux logiciels de création de PDF.

La suite bureautique LibreOffice (ou OpenOffice) permet de créer des PDF à partir de tout document, d'une façon similaire à une impression. Ceci est très facile et permet de faire des liens hypertexte et de définir la résolution des images. Problème : si le document n’a pas été conçu avec OpenOffice, il peut y avoir des défauts de mise en page ou de mise en forme. Microsoft a indiqué que sa prochaine version de Word permettra de faire du PDF nativement.

Le PDF permet de protéger des documents (important pour les documents en ligne). Il est possible d’autoriser ou non l’impression et la modification. La lecture peut être protégée par une licence

dans le cas des livres électroniques et des abonnements à des bibliothèques virtuelles par exemple. Les logiciels d'Adobe incorporent un système de gestion électronique des droits (DRM).

Il existe une version de PDF destinée spécifiquement à la chaîne graphique : PDF/X.

3.2. Traitements de texte

Quand on veut transmettre un document de façon qu'il puisse être modifié par son destinataire, il faut choisir un format de traitement de texte.

Traditionnellement on recommandait le format RTF, bien compris par la plupart des logiciels de traitement de texte. Aujourd'hui, il est plus efficace d'utiliser le format « Word 97/2000/XP », qui est plus riche et mieux interprété par les principaux traitements de texte que le format RTF. Pour échanger avec des utilisateurs disposant d'un traitement de texte propriétaire plus ancien, il faudra parfois choisir le format « Word 6 ».

Attention : nombre d'utilisateurs de versions plus récentes de *Microsoft office* ont tendance à enregistrer leurs documents dans un format trop récent pour être bien compris des autres logiciels (Word XML 2003 ou `docx`).

Les logiciels de la suite OpenOffice utilisent les formats ouverts ODF (open document format) : ODT, ODS... Pour l'instant ces formats ne sont pas encore lus par *Microsoft office*, il est donc préférable de les éviter quand on ne connais pas le logiciel employé par le destinataire. Ce format, porté par un consortium qui appellé OASIS, doit être présenté à l'ISO en vue de devenir un standard. La DMGE recommande ce format de fichier pour le traitement de texte (en utilisant OpenOffice).

Quand un document doit être travaillé à plusieurs mains, il peut être utile d'activer le suivi de révisions, qui permet de suivre les modifications apportées par chaque rédacteur.

3.3. Technologie web

Quand un document n'est pas destiné à être modifié (traitement de texte) et ne nécessite pas une impression (PDF) mais seulement une lecture à l'écran, le plus élégant/efficace est d'utiliser soit du texte brut (sans format, par exemple le corps d'un courrier électronique) soit un document HTML.

Quand le document présente un intérêt économique ou une importance particulière (newsletter, p.ex.), il peut être approprié de prévoir une feuille de style élaborée tenant compte des différentes situations de lecture, y compris les cas d'impression papier.

4. Images fixes

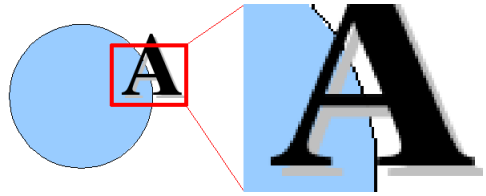
Comme pour les documents textuels, il faut distinguer, pour les images, les formats destinés au travail, à l'archivage et à la communication. Pour l'archivage, la règle est, le plus souvent d'utiliser les formats les plus proches de l'outil de création ou de numérisation. Pour les format de travail, il est recommandé de conserver le document dans le format spécifique au logiciel utilisé : PSD pour *Photoshop*, AI (EPS) pour *Illustrator*, FLA pour *Flash*, XCF pour *GIMP* etc. Ces formats, pour les logiciels les plus évolués du moins, incluent toutes les possibilités offertes par ces logiciels et permettent souvent de mêler sous forme de calques ou de canaux des bitmaps, des objets vectoriels, des chemins etc. Ce travail, à conserver autant que possible, peut ensuite être “exporté”, c'est à dire traduit, réencodé, dans un format autre, destiné à la publication visée.

Nous présenterons ici les principaux formats d'images, leurs avantages et leurs inconvénients. Le détail de ces formats et de leurs possibilités seront étudiés plus en détail au second semestre.

4.1. Bitmaps “sans” perte en “vraies” couleurs

Le type le plus usuel d'images numériques fixes est celui du *bitmap* ou *image en mode point*. Ces images sont constituées d'une grille ou matrice de rectangles de mêmes dimensions appelés *pixels*. Les principaux formats en mode point sont PNG (prononcer « ping »), JPEG et GIF, mais il en existe de très nombreux autres.

Figure 3.1. Agrandissements d'une image en mode point



Les formats de bitmap diffèrent tout d'abord dans leurs possibilité de rendu par le modèle de couleurs (et donc le nombre de couleurs) qu'ils autorisent. Le modèle de représentation le plus usuel des couleurs est appelé, improprement « vraies couleurs ». Il code ces couleurs, par synthèse additive (comme sur un écran lumineux), à l'aide d'une combinaison de rouge, vert et bleu, à raison de 8 ou 16 bits (1 ou 2 octets) par composante et par pixel, soit 16 M teintes (ou $2{,}8.10^{14}$ teintes, pour les archives et traitements professionnels).

Figure 3.2. Synthèse additive

On appelle généralement *profondeur des couleurs* le nombre de bits réservés pour chaque pixels. Si l'on utilise 8 bit par composante, la profondeur des couleurs est donc de 24 bpp (bits par pixel). Dans ce cas chaque composante est représentée par un nombre de 0 à 255, soit deux chiffres hexadécimaux :

hexadécimal	00	33	66	99	CC	FF
décimal	0	51	102	153	204	255
proportion	0 %	20 %	40 %	60 %	80 %	100 %

On utilise maintenant souvent la notation suivante, issue du *web* : #3399CC, qui signifie : rouge à 20 %, vert à 60 % et bleu à 80 %.

Notons que le format PNG prévoit également la possibilité d'une quatrième composante, d'opacité : RGBA (rouge vert bleu alpha), soit 32 bpp. Ceci est utilisé par les systèmes d'exploitation pour les icones et pointeurs et maintenant très répandu sur le *web*, même s'il n'est pas toujours bien pris en compte par certaines version du navigateur *Internet Explorer*.

Il existe d'autres modes de représentation des couleurs, par exemple TSL (teinte, saturation, luminosité) pour la vidéo ou CMJN (cyan, magenta, jaune, noir) pour l'imprimerie, puisque les encres fonctionnent par synthèse soustractive (le format TIFF, qui autorise ce modèle, est dominant dans l'imprimerie).

Figure 3.3. Synthèse soustractive

Une possibilité de codage des bitmaps consiste à donner la liste des teintes pour les différents pixels, les uns à la suite des autres. C'est le format brut, il est extrêmement consommateur de ressources. Les formats récent utilisent, *a contrario*, des algorithmes de compression de données pour réduire la taille des images (par différentes méthode repérage de redondances). Ainsi une image de 2560×1712 pixels en 24 bit/px, qui pèse 12,5 Mio en format brut, ne pèsera en PNG que de 16kio (uni) à 6 Mio (photo) à 12,5 Mio (aléatoire). Ce type de format est appelé “sans” perte : il

y a bien des pertes au moment de la numérisation mais pas du fait du codage du bitmap sous forme de fichier. Un même fichier a donc la même interprétation partout : l'image est conforme avec ses visualisations dans les logiciels de dessin.

Ces formats sont parfait pour le texte et les petites images.

4.2. Bitmaps “sans” perte en couleurs indexées

Quand une image comporte très peu de couleurs (256 ou moins), par exemple un dessin avec de grandes surfaces monochromes, les formats en vraies couleurs sont trop gourmand en quantité d'information. Il est, dans ce cas, plus efficace d'utiliser une « palette », un index des teintes d'une image, de façon à coder chaque pixel sur 8 bits (au lieu de 24 ou 32). Premier bénéfice, immédiat : une réduction de la taille par 3 (ou 4) au moins.

En pratique, les formats qui utilisent des couleurs indexées emploient des algorithmes qui tirent parti de similarités entre parties de l'image, similarité dont la probabilité augmente quand le nombre de couleurs diminues, particulièrement en cas d'aplats de couleur unis. Le coefficient de réduction dépasse donc souvent le facteur 3. Cas hypothétique extrême : une image de 2560×1712 pixels entièrement monochrome ne pèse que 640 o. Avec le drapeau japonais (aplat très simple), on n'atteint que 23 kio, soit 500 fois moins que le format brut.

Le format sans perte en couleurs indexées le plus populaire, bien qu'obsolète, est le format GIF. GIF permet de disposer d'une “couleur” entièrement transparente. Le format généralement le plus adapté, aujourd'hui, est le format PNG (qui permet donc de coder une image soit en vraies couleurs, soit en couleurs indexées). En format PNG on dispose jusqu'à 256 niveaux de transparence, pour chaque teinte possible.

Cas particulier de palette : la palette “web”. Elle est principalement composée de couleurs dont les composantes RGB sont des multiples de 20%. En hexadécimal, les couleurs s'écrivent donc comme une suite de l'un des six nombres suivants : 00, 33, 66, 99, CC, FF (en décimal : 0, 51, 102, 153, 204, 255). À ces 6×6×6=216 couleurs s'ajoutent le transparent et huit couleurs de base, en tout 225 teintes (voir le cours *Documents structurés pour le web* pour plus de détails).

4.3. Bitmaps avec pertes

Les formats sans perte sont généralement peu adaptés aux photographies. Tout d'abord parce que l'on n'a le plus souvent pas besoin d'une telle exactitudes, des pertes raisonnables sont admissibles. Ensuite parce que les algorithmes de compression de PNG et GIF ne sont pas très efficaces sur les images photographiques. JPEG est un format d'image permettant de compresser plus de telles images. La perte de qualité utilisée par JPEG tire partie d'une redondance psychovisuelle.

Redondances de code : Il y a redondance dans le code d’une image si l'on peut utiliser moins de symbole pour la coder que strictement nécessaire. Les formats d'images peuvent utiliser des algorithmes génériques (pas adapté à un type de document en particulier), comme l'algorithme LZW utilisé par GIF et TIFF, pour compresser des images.

Redondances inter-pixels : Les images réellement utilisées ne sont pas des arrangements aléatoires de teintes. On peut repérer des corrélations statistiques entre pixels proches, par exemple entre une ligne et la suivante. Ce type d'analyse permet de définir des algorithmes de compression spécifiques aux images, tels que pour PNG. Par exemple, on n'est pas obligé de coder une ligne entière mais seulement des différences avec la précédente.

Redondances psychovisuelles : Quand nous observons une image, surtout de résolution fine, nous ne regardons pas tous les pixels. Seuls certains éléments clés sont analysés, par exemple des changements de couleurs, de contraste ou des variations à grande échelle. Les autres informations sont dites psychovisuellement redondantes (cette notion est relative puisqu'un même observateur peut oserver une image avec plus ou moins de détail). Éliminer ces informations permet d'alléger le fichiers, au prix d'une perte de qualité de l'image. Les données perdues le sont, le plus souvent, de façon irréversible, contrairement à ce qui se produit dans de nombreux films policiers…

Toutefois, supposer cette redondance n'est pas toujours à propos :

- La compression JPEG maltraite les bords francs.
- JPEG n'est pas le format le plus compact pour les images constituées d'aplats de couleur.
- JPEG est bien adapté à la photographie numérique (on peut souvent se contenter d'un bas niveau de qualité JPEG).
- JPEG est rarement le meilleur choix pour les images de petite taille (on est alors obligé de choisir un haut niveau de qualité JPEG).

JPEG permet également d'inclure des données sur la prise de vue (marque de l'appareil, optique, réglages utilisés, orientation, etc.).

4.4. Résumé synoptique sur les bitmaps

Le choix entre PNG, GIF et JPEG peut se synthétiser avec la grille suivante :

Compression “sans perte”, avec palette	Compression sans perte, “vraies” couleurs	Codage avec pertes
PNG en mode palette (ou GIF)	PNG “vraies” couleurs (*truecolors*)	JPEG
La perte se fait en choisissant une palette de 256 couleurs (au plus). Plus la palette est réduite, moins le fichier sera volumineux.	Pas de perte liée au codage.	On choisit le niveau de perte (donc la qualité de l'image résultante) de façon à ajuster la taille de fichier.
indiqué quand il y a peu de couleurs (<256)	16 M couleurs (RVB)	16 M couleurs (RBV)
Parmi les couleurs il y a la transparence (à préférer pour les logotypes). En PNG, on peut même avoir 256 niveaux d'opacité (l'alpha).	256 niveaux d'opacité (alpha) - utile pour représenter des ombres ou des superpositions	pas de transparence possible
GIF : possibilités de petites animation (pas pour PNG) - peu professionnel (utiliser plutôt Flash)	image fixe	image fixe
indiqué pour les petites images (miniatures, logos, etc.) ou quand il y a peu de couleurs (<256)	indiqué pour les images de taille moyenne demandant plus de 256 couleurs	indiqué pour des photographies (sauf en cas de petite taille)
traite bien les bords francs et les aplats de couleur (ex.: texte, dessins en ligne claire)	traite bien les bords francs et les aplats de couleur (ex.: texte, dessins en ligne claire)	maltraite les bords francs et n'est pas efficace sur les aplats de couleur

Voici les principaux noms des dimensions (en pixels) des formats vidéo usuels (ci-après). On sera vigilant les notions de rapports “4/3”, “16/9” ou “16/10”, dans la mesure où dans le rendu de ces formats, les pixels ne sont pas nécessairement carrés (en particulier dans le cas des formats télévisuels hérités d'avant l'ère numérique).

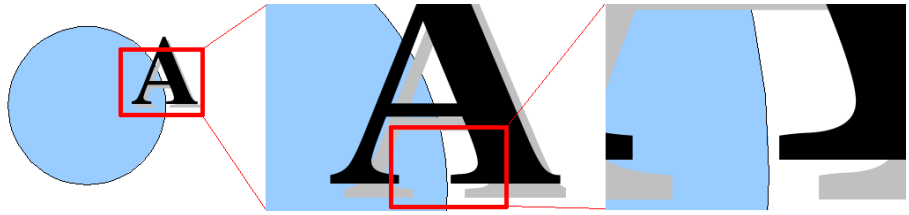
Dimensions (px)	Nb. pixels	Nom, utilisation
320 × 240	64 k	QVGA (quart de VGA), téléphones, webcams
640 × 480	300 k	VGA (video graphics array), anciens moniteurs de 14" ou moins, téléphones, PDA

Dimensions (px)	Nb. pixels	Nom, utilisation
720 × 480	340 k	NTSC (SDTV480i ou EDTV480p)
720 × 576	400 k	PAL/SECAM (SDTV)
800 × 600	0,5 M	traditionnellement conseillé pour un moniteur de 14" à 15" (bas de gamme), un des formats SVGA
1024 × 768	0,8 M	XGA (extended graphics array), traditionnellementconseillé pour un moniteur de 17" (moyenne gamme), un des formats SVGA
1280 × 720	0,9 M	HD TV 720 (ou WXGA-H)
1366 × 768	1,0 M	WXGA (wide XGA), moniteur 17" 16/9
1440 × 900	1,3 M	format fréquent pour les portables 17" dits 16/9
1280 × 1024	1,3 M	SXGA (super XGA), traditionnellement conseillé pour un moniteur 19" (haut de gamme), un des formats SVGA
1600 × 1024	1,6 M	WSXGA (wide SXGA), moniteurs 19" dits 16/9
1680 × 1050	1,8 M	WSXGA+, moniteur 19" dits 16/9
1600 × 1200	1,9 M	UXGA (ultra XGA), conseillé pour un moniteur 21" 4/3 (haut de gamme), un des formats SVGA
1920 × 1080	2,0 M	HDTV 1080
1920 × 1200	2,3 M	WUXGA (wide UXGA), moniteur 21" dit 16/9

4.5. Dessins vectoriels

L'image n'est pas constituée d'une grille de pixels mais est de nature symbolique. Il s'agit d'une succession d'objets graphiques : aplats de couleur (uni, dégradé, texture...), lignes, texte. Ces objets sont définis par des fonctions géométriques, des coordonnées, des teintes et autres paramètres d'aspect. Ils sont vectoriels : ils peuvent être multipliés par un nombre servant de facteur d'échelle (un scalaire) sans perte de qualité. Parfois les dessins vectoriels incorporent également des bitmaps... ils ne sont donc plus a proprement parler vectoriels, même si le format de fichier reste vectoriel.

Figure 3.4. Agrandissements[2] d'une image vectorielle



Principaux formats :

[2]Attention : cet exemple est réalisé à l'aide d'une image en mode point. Son affichage ni son impression ne peuvent donc rendre correctement l'effet d'un agrandissement vectoriel, seulement le suggérer.

- SVG : pour le web, principalement (interactivité possible)
- Flash : dessins animés et publicité web, permet d'ajouter de l'interactivité par programme
- EPS (ou AI) : format natif d'Adobe Illustrator, le standard professionnel
- WMF (ou EMF) : métafichier windows, utilisé principalement pour des “cliparts” destinés aux traitement de texte

Chapitre 4. Structure d'un ordinateur

1. Les parties constitutives d'un ordinateur

Le mot « ordinateur » a été forgé pour la société *IBM France* en 1955. À l'époque le mot « *computer* »/« calculateur » était réservé aux calculateurs numériques, principalement destinés aux calculs scientifiques ; IBM cherchait un nom pour désigner plus généralement une machine à traiter l'information[1]. François Girard, alors responsable du service promotion générale publicité eut l'idée de consulter son ancien professeur de lettres à Paris. Jaques Perret alors professeur de philologie latine à la Sorbonne, proposa le 16 avril 1955 le mot « ordinateur » en précisant que le *Littré* l'indiquait pour signifier « Dieu qui met de l'ordre dans le monde » mais que « ce mot est tout à fait sorti de l'usage théologique ».

Figure 4.1. Un ordinateur personnel : le Superbrain II (1983) (DR[2])

L'*unité centrale* : l'ordinateur proprement dit, ce qui tient dans le boîtier principal.

Les périphériques, ce qu'il y a "autour" : de sortie : écran, imprimante ; d'entrée : clavier, souris, scanner,... L'organisation physique précise peut varier : écran associé à l'unité centrale ou pas, disques durs internes ou externes, clavier ou non, etc. On parle donc de *périphériques* même pour des éléments physiquement placés dans l'unité centrale : les périphériques sont alors les éléments qui ne sont pas nécessaires à un ordinateur pour fonctionner.

Parfois la définition précise de ce qui constitue "un" ordinateur n'est pas évidente. Ainsi certains serveurs sont constitués de plusieurs « lames », chacune étant un véritable ordinateurs, enfichées dans une même unité "centrale". La limite n'est donc plus tranchée entre ordinateur et ferme d'ordinateurs.

N'oublions pas, non plus, que beaucoup de téléphones cellulaires et d'assistants personnels numériques sont, en fait, des ordinateurs plus puissants que les PC d'il y a seulement quelques années. Le mot « ordinateur » est donc plus un terme fonctionnel qu'un terme technique.

[1] À cette date, IBM avait déjà sorti un premier « calculateur », destiné au calcul scientifique. Il s'agissait alors de sortir un premier « ordinateur » destiné aux entreprises.
[2] Merci à Dave Dunfield (DR) : http://www.parse.com/~ddunfield/museum/

Rappel de la loi de Moore (1965, 1975) : le nombre de transistors des processeurs double à coût constant tous les deux ans. Cette loi est assez bien vérifiée empiriquement. On l'exprime souvent sous une forme simplifiée d'un doublement de puissance brute des ordinateurs (mémoire, capacité des disques, puissance des processeurs) tous les 18 mois. Cette seconde forme est très grossière, donc invérifiable à proprement parler, mais donne un bon ordre de grandeur. Dans les deux cas, il s'agit d'une progression exponentielle, donc extrêment rapide. Si l'on prend l'hypothèse d'un doublement tous les deux ans, on obtient un facteur 8 pour 6 ans, 32 pour 10 ans et 181 pour 15 ans et 1024 pour vingt ans. Les spécialistes prévoient un fort ralentissement de la progression, voire une limite, des techniques actuelles dans 5 à 10 ans, principalement pour des raisons physiques.

2. Les éléments centraux usuels

L'*alimentation électrique*. Elle est parfois complétée ou remplacée par une batterie, pour un portable, ou un onduleur, pour une machine fixe. Un onduleur est essentiellement une batterie externe. Les serveurs (ou les grappes de serveurs) sont souvent dotés deux alimentations, pour limiter les risques en cas de défaillance.

La *carte mère* est un circuit imprimé sur lequel sont soudés ou enfichés les divers composants de l'unité centrale. Elle est munie de certains connecteurs réservés (socket ou slot pour le processeur, bus AGP pour les cartes graphiques, connecteurs spéciaux pour les lecteurs de disques...) et de connecteurs génériques pour les cartes de périphériques : ISA (vieille norme PC), NuBus (vieille norme Mac), PCI... Les cartes connectées à la carte mère sont appelées *cartes filles*.

La *mémoire centrale*, composée de RAM (*random access memory* : mémoire à accès aléatoire) ou *mémoire vive* et de ROM (*read only memory* : mémoire en lecture seulement, cette formulation n'est, en fait, plus appropriée) ou *mémoire morte*. La mémoire vive sert à stocker l'information en cours de traitement. La mémoire morte contient des programmes qui permettent de définir la configuration permanente de l'ordinateur (*setup*, la configuration étant stockée dans une mémoire particulière appelée CMOS) et qui permettent de démarrer l'ordinateur, charger le système d'exploitation et gérer les fonctionnalités de base de l'infrastructure matérielle (BIOS). Les deux types de mémoire ont un fonctionnement similaire : lire et écrire dans des cases (généralement de 4 ou 8 octets) repérées chacune par une adresse (généralement de 32 ou 64 bits). Les RAM sont généralement des mémoires très rapides mais qui perdent leurs données quand elles ne sont plus alimentées électriquement. Les mémoirees mortes ne sont souvent plus aujourd'hui réellement des ROM : ce sont souvent des mémoires "flash" similaires à celles des clés USB et cartes mémoire.

Le ou les *processeurs centraux* (ou CPU : *cental processing unit*), qui effectuent les traitements principaux de l'information. Aujourd'hui la plupart des ordinateurs personnels comportent plusieurs processeurs regroupés sur la même puce. Les ordinateurs comportent généralement plusieurs processeurs secondaires : processeur graphique, contrôleur de disque etc. ; ceux-ci sont considérés comme périphériques. La notion de "puissance" d'un processeur n'a *aucun sens* dans l'absolu. On peut au mieux compter la fréquence de l'horloge qui cadence ses travaux (en GHz, gigahertz), mais cette quantité n'a de signification qu'à l'intérieur d'une famille de processeurs donnée.

Figure 4.2. Le mythique Z80 (une des nombreuses version) (DR)

Pour accélérer l'accès des processeurs aux données en mémoire, les ordinateurs sont souvent dotés de systèmes de *mémoire cache*. Ce système est composé de deux à trois niveaux de cache de taille croissante et de vitesse décroissante. Le cache de niveau 1 (L1), intégré au processeur, est très rapide mais peu volumineux (pour des raisons de coût), celui de niveau 2 (L2), plus lent est dans le processeur ou sur la carte mère. Quand il y en a un, le cache de niveau 3 est sur la carte mère.

3. Les périphériques usuels

Le clavier. Les claviers français sont souvent nommés AZERTY (d'après les touches de la première ligne), les claviers anglais sont nommés QWERTY. Ces dispositions ont été conçues à l'époque des machines à écrire mécaniques pour ralentir la frappe : des frappes trop rapides faisaient se toucher… et se coincer les marteaux.

Le dispositif de pointage : souris, écran tactile, trackball, tackpoint, touchpad etc.

La *“carte” graphique*, qui comporte elle-même un ou plusieurs processeurs et sa propre mémoire vive, généralement très rapide. Elle est parfois intégrée à la carte mère ; on ne devrait donc pas parler de « carte graphique » mais de « système graphique ». Aujourd'hui l'image affichée à l'écran est calculée par la carte graphique : on a donc deux séries de processeurs (graphiques et centraux) qui se partagent une partie de la mémoire.

Le *moniteur* (l'“écran”). Pour la conception web ou multimédia, il n'est pas rare d'utiliser deux moniteurs : un pour la conception, un pour la consultation du résultat.

La *mémoire de masse* : disque(s) dur(s), disquettes, CD, DVD, cartes mémoire, clefs USB etc. Toute forme de stockage de l'information au delà du traitement en cours.

Autres périphériques usuels : “carte“ son, “carte” réseau (ethernet, WiFi…), imprimante, webcam, scanner, modem…

Chapitre 5. Exploitation

1. Cadre général - systèmes et environnements

Un ordinateur est capable de faire tourner différents programmes ou logiciels rendant divers services. Un ensemble logiciel coordonné assurant divers services de même type est appelé une *application*. Le traitement de texte, le tableur, la gestion de l'e-mail, etc. sont des applications.

Chaque application opère (on dit « tourne ») sur un ordinateur en utilisant ses différents composants, qui lui sont accessibles grâce au s*ystème d'exploitation* (en anglais : OS, *operating system*). Les applications modernes présentent (on dit « affichent ») des éléments graphiques à l'écran et réagissent aux ordres que l'utilisateur donne grâce aux périphériques d'entrée : clavier, souris ou autres. Les éléments graphiques d'une application apparaissent généralement dans une ou plusieurs zone, généralement rectangulaires appelées fenêtres.

Les principes qui régissent les fenêtres dépendent d'une application particulière appelée *système de fenêtre*. Chacun des principaux systèmes d'exploitation a le sien, mais les principes généraux sont les mêmes.

2. Programmes et processus

L'être humain qui doit utiliser une application n'interagit que d'une façon virtuelle avec celle-ci : en réalité il interagit matériellement avec la machine. Il faut donc une interface entre les deux : elle est réalisée par le système d'exploitation, aussi appelé système (tout court), qui abstrait la machine.

Le système donne des moyens logiciels permettant aux diverses applications d'utiliser chaque élément matériel. Par exemple pour un disque dur, le système contient un gestionnaire de disque dur, qui permet aux applications d'enregistrer des données sans s'inquiéter des différents modèles de disques. Les éléments du systèmes qui donnent accès aux périphériques sont appelés *pilotes* ou *gestionnaires*, en anglais *drivers*.

La contrepartie en mémoire d'un logiciel en train de s'exécuter est appelée *processus*. Le système a notamment pour rôle de gérer la mémoire et le temps de calcul, notamment pour le répartir entre les divers processus (c'est à dire entre les tâches à effectuer à un moment donné). Quand un seul processus est réellement actif à un moment donné ou que les processus se passent la main (ex. : windows 95) et, donc, sont susceptibles de bloquer la machine, on dit que le système est *monotâche*. Quand c'est le système qui découpe arbitrairement son temps et le répartit entre les processus, on dit qu'il est *multitâche*.

Les principaux types de systèmes actuels sont : UNIX (dont Linux), WindowsNT (dont Windows XP), MacOS, DOS (associé à Windows non-NT). Les systèmes "récents" (Unix, WindowsNT, MacOS depuis la version 9) sont réellement multitâches et sont d'architecture générale relativement proche.

Les services permanents (serveur web, serveur de fichiers, etc.) sont assurés par des processus "dormants" qui se "réveillent" quand une requète arrive. On parle parfois de *démons*. Sous WindowsNT les services automatiquements lancés sont contrôlés par un *gestionnaire de services*. Sous Unix (dont Linux), les outils de configuration propre à chaque distribution permettent de décider quels services doivent être ou non activés.

Les systèmes évolués disposent d'un (ou plusieurs) *gestionnaires de tâches*, qui permettent de gérer certains privilèges ou de "tuer" des processus (parce qu'ils ne répondent plus, par exemple). Sous les systèmes WindowsNT, on l'obtient par **#^Suppr** (contrôle+alt.+suppr.).

3. Gestion des utilisateurs

En informatique, le mot « *utilisateur* » désigne rarement une personne en chair et en os, mais plutôt un *compte* informatique : un *nom d'utilisateur* ou *logname* ou *login* muni d'un *mot de passe* (qui doit

être changé régulièrement). Chaque compte dispose de certains *privilèges*. En général sur les Unix et dérivés un compte particulier, appelé *root* (litt. "racine") ou *administrateur*, a tous les privilèges et permet d'administrer une machine ou un réseau.

Note : Un mot de passe ne doit être ni nom commun ni nom propre et doit être suffisamment long (10 caractères minimum) et, autant que possible, mêler des chiffres et des lettres, voire d'autres caractères. Sous windows NT on change son mot de passe via `#^Suppr`.

Les multitâches peuvent être également multi-utilisateurs. Quand un système cloisonne de façon étanche ce qui appartient (espace disque, bases de données, configuration, etc.) à différents utilisateurs, on dit qu'il est *multiutilisateur*. C'est le cas des Unix et dans une assez large mesure des WindowsNT (s'ils sont configurés pour cela). Dans ce cas, chaque processus appartient à un compte (son propriétaire), ce qui lui confère le droit d'effectuer certaines actions et d'autres pas : autorisation d’écrire dans tel ou tel répertoire, droit de faire telle ou telle action particulière sur une machine,...

Cette séparation des privilèges est essentielle dans la lutte contre les *virus* : ce qui affecte un compte ne peut pas toucher un autre compte ou les logiciels proprement dit. Toutefois, ceci n'est opérant qu'à condition de disposer de plusieurs comptes séparés munis de privilèges définis et de ne jamais travailler en tant qu’administrateur.

Toutefois, ce type de stratégie est insuffisante contre les *vers*, qui infectent des processus permanents (des serveurs, web ou autres) et se propagent à travers les réseaux.

4. Systèmes d'exploitation

La clé des systèmes d’exploitation modernes est la modularité (division des tâches). Le principe est de séparer ce qui relève des processus de ce qui relève de l'accès aux ressources et de leur administration. Le système abstrait toutes les ressources de la machine au bénéfice des processus. Inversement, il présente pour la machine une interface commune aux logiciels. Dans les systèmes les plus étanches (les mieux conçus) les processus n'ont aucun accès direct aux ressources et doivent passer obligatoirement par le système.

La mémoire est une ressource, donc : un processus n’écrit jamais directement dans la mémoire (il passe toujours par le système d’exploitation).

Les processus eux-mêmes sont des ressources, donc : si des processus veulent communiquer entre eux, ils doivent passer par l’intermédiaire du système d’exploitation (cette pratique n'est pas toujours systématique, malheureusement).

Un système d'exploitation comporte deux grandes sous-parties : 1) le *noyau* et les pilotes associés : contrôle des éléments, périphériques, tâches, etc., 2) les *bibliothèques* [*librairies*] standards : les traitements communs à de nombreux programmes (procédures génériques ou liées au système).

Grandes familles de systèmes pour ordinateurs :

- *Unix*, dont *Linux*, chacun étant régulièrement mis à jour et diffusé sous la forme de distributions. Les Unix récents gèrent plusieurs processeurs sur une même machine voire plusieurs machines agglutinées en "clusters".
- *Windows NT* : inspiré d'Unix, dont il tend à se rapprocher en conservant une ergonomie par fenêtres.
- DOS + *Windows* : système personnel de Microsoft, aujourd'hui remplacé par la famille WindowsNT.
- *OS/2* : l'alternative à Windows conçue par IBM, aujourd'hui disparue.
- *MacOS* (avant la v. 9) : système du constructeur Apple pour ses Macintosh.
- MacOS (depuis la v. 9) : inspiré d'Unix, mais conservant l'ergonomie Macintosh.

La gestion des fenêtres est dévolue à un (ensemble de) programme particulier appelé *système de fenêtres* (sous Unix : *X window* ou X11). Leur habillage, déplacement, redimentionnement, etc. est dévolu au *gestionnaire de fenêtres* (nombreux sous Unix, les principaux : KDE, Gnome). A l'idéal ces éléments ne devraient pas être dans le système. C'est (malheureusement) largement le cas pour MacOS et Windows.

5. Systèmes de fichiers

5.1. Généralités

Un *disque* est une unité physique sur laquelle il est possible d'enregistrer des informations. Dans les systèmes on emploie généralement cette terminologie même quand il n'y a pas à proprement parler d'objet circulaire plat en jeu (ex.: clef USB).

Ces informations sont regroupées dans des *systèmes de fichiers* (en anglais FS, *filesystems*). Ces systèmes de fichiers peuvent occuper un disque entier (c'est le cas général pour les périphériques amovibles : disquettes, clef, CD/DVD...) ou bien être contenus dans une ou plusieurs *partitions*, lesquelles sont simplement des segment du disque (c'est généralement le cas pour les disques dur). Les partitions sont parfois appelées *disques logiques* et sont alors traitées par le système comme s'il s'agissait de disques différents. Préparer un disque pour y mettre un système de fichier, s'appelle *formatage de bas niveau*. Mettre un système de fichier sur une partition ou un disque est appelé *formatage* (ou *formatage de haut niveau*). Cette double terminologie, très largement en usage, ne se justifie plus vraiment aujourd'hui et est parfois source de confusions.

Les systèmes de fichiers sont des ressources parmi d'autres. On peut imaginer un système sans disque (OS des routeurs, p.ex.). À l'inverse, un OS donné peut gérer de nombreux types de systèmes de fichiers : tout dépend des gestionnaires intégrés au noyau. Ainsi les WindowsNT récents lisent les disques Macintosh (HFS). De même Linux lit tous les systèmes courrants (dont Macintosh et Windows). Les systèmes de fichiers Unix s'autodéfragmentent à l'utilisation. Les autres doivent être régulièrement remis "à neuf" (défragmentation voire reformattage).

La structure d'un système de fichiers est une *arborescence* : elle est constituée d'un dossier *racine* qui, lui-même, contient des *dossiers* ou *répertoires* qui eux-même contiennent etc. C'est, finalement, un arbre dont les "noeuds" sont les dossiers et les "feuilles" les fichiers (et autres nœuds spéciaux). Chaque dossier peut contenir des fichiers éventuellement de plusieurs types différents.

Pour que le système de fichier d'une partition soit utilisé par le système d'exploitation, il faut que cette partition soit *montée*. Cette opération est, le plus souvent automatique. En revanche, le démontage doit généralement être demandé manuellement pour les systèmes amovibles (clef USB, disquette etc.). Comme une information n'est pas toujours transmise immédiatement aux systèmes de fichiers montés par le système (c'est ce qu'on appelle un fonctionnement asynchrone), il est nécessaire de toujours éteindre un ordinateur par la procédure prévue, de façon que les systèmes soient correctement "démontés". Sinon, des erreurs peuvent apparaître.

Note pour Unix : Sous Unix, tous les systèmes de fichiers sont *attachés* (ou montés) en remplacement d'un dossier particulier de l'arborescence déjà montée ou directement à la racine. Attention : plusieurs chemins dans l'arbre peuvent désigner exactement le mêmes fichier (au même emplacement) sur un disque. On dit que ces chemins sont des *liens physiques* vers le même fichiers. Dans ce cas, l'arborescence n'est pas vraiment un arbre puisque deux "branches" convergent.

(à faire : schéma)

Note pour MacOS : Sous MacOS les systèmes de fichiers apparaissent généralement sous la forme d'un dossier sur le bureau (le plus souvent avec un icône de disque, par défaut).

Note pour Windows : Sous Windows les systèmes de fichiers apparaissent sous la forme d'une lettre sur le poste de travail : `A:` pour le système sur la disquette, `C:` pour le "disque" - en fait le système de fichiers - principal, `D:` pour le suivant, etc.

Sous les environnement de travail usuels un fichier peut se contenter de pointer vers un autre fichier (ou dossier), on parle de *raccourci* (ou *alias* sur MacOS). Sous Unix, il existe une possibilité supplémentaire : les *liens symboliques*. C'est une version renforcée de l'alias : éditer un lien symbolique revient à éditer son fichier cible (sauf précaution particulière). Ceci peut être particulièrement utile aux serveurs *web* et aux serveurs de fichiers (FTP),... à condition de prendre garde à ne pas confondre le lien avec sa cible.

Sur les serveurs en réseau, les utilisateurs ou les groupes d'utilisateurs se voient généralement imposer un *quota* d'utilisation des disques. Un disque leur apparaît plein quand le quota (en volume et/ou en nombre de fichiers) est atteint.

Les gros systèmes ou les ordinateurs qui demandent une forte sécurisation des données peuvent mettre en place un système de *journalisation* qui garde trace de toutes les opérations et permet de récupérer des fichiers "accidentés". Les ordinateurs serveurs sont, par ailleurs, souvent équipés de systèmes de réplication de disques (RAID) qui permet de conserver les données même quand l'un des disques durs tombe en panne.

5.2. La mémoire virtuelle

Quand une machine ne dispose plus d'assez de mémoire vive pour y placer tous les processus, elle peut utiliser une partie (délimitée) de l'espace disque appelée *mémoire d'échange* (en anglais *swap*). La combinaison de la mémoire vive et de la mémoire d'échange pour y placer les processus, de façon transparente pour eux, est appelée *mémoire virtuelle*. Attention : comme l'accès aux données sur un disque est considérablement plus long qu'en mémoire vive, cela peut ralentir beaucoup une machine. Ce ralentissement est particulièrement sensible quand plusieurs processus trop "volumineux" sont utilisés simultanément (et non consécutivement) : la machine passe alors son temps à transférer les processus depuis la mémoire d'échange vers la mémoire vive et inversement. Il faut donc veiller à dimensionner correctement la mémoire vive, la mémoire d'échange ne devant être qu'une roue de secours.

Ceci est d'autant plus important pour les serveurs *web* (ou de fichiers) tournant sous Unix. En effet, les systèmes Unix conservent un fichier lu récemment en mémoire vive (tant qu'il y en a de libre). Un serveur *web* a donc intérêt à avoir beaucoup de mémoire : dans ce cas les fichiers sont lus à partir de leur copie (cache) en mémoire vive et non à partir du disque. Cela limite les accès au disque et peut augmenter considérablement sa vitesse de traitement, même pour de gros sites.

5.3. Privilèges sous Unix

Sous Unix, cas le plus fréquent de serveurs web et FTP, les privilèges d'accès à chaque fichier et à chaque dossier se répartissent en trois catégories :

- U - Les privilèges pour le compte utilisateur propriétaire du fichier. Généralement, le propriétaire d'un fichier est celui qui l'a créé.
- G - Ceux pour un groupe d'utilisateurs bien défini appelé « groupe propriétaire ». Le groupe "propriétaire" est l'un des groupes d'utilisateur choisi par le propriétaire.
- O - Ceux pour le reste des utilisateurs (o : *other*).

Pour chacune de ces catégories, sont définit trois bascules :

- R - Autorisation de lire le fichier ou de lister le dossier.
- W - Autorisation de modifier le fichier ou le dossier.
- X - Pour un fichier : autorisation d'exécuter (c'est un logiciel). Pour un dossier : autorisation d'accéder au contenu.

Le propriétaire d'un fichier ou dossier, ou l'administrateur, peuvent donc régler neuf autorisations indépendantes.

Index

T

U

V

W

X

Y

Z