ANNEXE 5

ECHANGE DE FICHIERS

Un fichier créé par un programme donné (logiciel, *application* ...) peut toujours être repris et modifié avec cet exécutable. C'est moins facile avec un autre logiciel ou avec ce même logiciel dans une autre version ou sous un autre système d'exploitation (SE, on dit aussi *plate-forme*...). Par exemple, si l'on a créé un document avec Word5 sous Dos, il semblerait normal de pouvoir le reprendre avec Word sous Windows ou même sous Macintosh. La même question peut se poser pour des *feuilles de calcul*, des bases de données, des dessins ... Nous allons donner quelques indications sur les différents obstacles à surmonter. Il faudra franchir en fait cinq barrières successives : celle du **support matériel**, celle de la **décompression** éventuelle, celle du format des **nombres**, celle du **codage du caractère** et celle de la **compatibilité logicielle**. On citera des solutions permettant un échange plus ou moins complet (RTF, PostScript, Acrobat, TIFF, JFIF, GIF).

1 - Compatibilité physique

Les informations sont codées de différentes façons selon le support par lequel elles sont véhiculées, disques de toutes natures, liaison électrique, ...

a - Disques. Sur ce genre de support, le codage du bit varie d'une "norme" à l'autre. De plus, ces supports contenant un grand nombre de fichiers, une **table** décrit la façon dont ils sont répartis sur le disque. Cette table (la FAT sous Dos) est propre à chaque SE. La compatibilité entre SE serait quasi nulle s'il n'existait pas des *utilitaires* pour adapter les disques d'un SE à un autre. L'utilitaire le plus connu est le logiciel *Apple File Exchange* (AFE), qui permet à un Macintosh sous System-7 de lire une disquette Dos (à partir de la version 7.5, AFE est appelé automatiquement quand le Mac reconnaît une disquette Dos). De même, Linux sait lire les disques et disquettes Dos (signalons qu'on peut faire cohabiter sur un même disque des partitions Dos et des partitions Linux).

Le Dos peut utiliser des utilitaires analogues, mais ils sont moins surs et peu employés. Depuis Dos7.1 (ou Windows95.0b), Dos utilise une FAT32 (32 bits) au lieu de la FAT16 classique (laquelle ne peut adresser que 65526 UA – cf. chap 8, §3 – et donc limite la taille d'une partition à 2 Go avec des UA de 32 Ko). Cette FAT32 (UA de 4 Ko possibles, pas de limitation en taille des partitions) n'est pas compatible avec les disques et les utilitaires en FAT16 (mais Linux depuis peu sait lire la FAT32).

En ce qui concerne les disquettes ou cassettes de haute capacité, la compatibilité est la même, donc plutôt faible. Une compatibilité intéressante : les lecteurs au nouveau format LS120, conçus pour traiter des disquettes 3,5" de haute capacité (120 Mo), peuvent lire les disquettes classiques (3,5", 1,44 ou 0,7 Mo) formatées pour ce même SE.

Les disques optiques non réinscriptibles (ou CD-Rom) offrent plus de compatibilité s'ils respectent la norme ISO 9660, qui les rend indépendants du SE au niveau physique (et même à tous niveaux sauf au niveau logiciel). Un lecteur de disque optique peut également lire les disques audio numériques (CD, *disque laser*), dont la machine peut jouer la musique, même en tâche de fond.

b - Signaux électriques. Les informations peuvent parvenir par voie électrique, optique ou par radio ; elles sont toujours converties en signaux électriques avant d'accéder au calculateur. Les conducteurs peuvent être ceux d'un réseau local, ou bien ceux du téléphone ordinaire (dit *commuté*) via un modem ou via Numéris, ou bien ceux accédant à une connexion (une *porte*) série ou parallèle. Les accès parallèles sont peu normalisés (sauf GPIB, totalement *portable*). Les entrées-sorties série se conforment à de nombreuses normes, de plus en plus performantes. Ces normes, si elles sont respectées, garantissent une **totale compatibilité** (au niveau physique) quel que soit le type de machine. Malheureusement, le monde DOS privilégie la norme RS232 et le monde Mac la norme RS485, du moins pour les cartes installées en usine (il faut insérer un adaptateur électronique très simple, ces normes ne différant que par des niveaux électriques). Pour les communications en **réseau**, la carte ou le modem, propre au type d'ordinateur et au réseau, garantit la compatibilité. Pour résumer, la compatibilité au niveau physique est toujours assurée quand l'information est transmise par voie électrique.

2 - Décompression

Une fois le fichier rendu accessible par l'un des moyens ci-dessus, il sera très souvent nécessaire de le décomprimer et même de le **désarchiver** (i.e. le scinder en plusieurs fichiers, tel qu'il était à l'origine). Outre des méthodes en développement (comme celle de type *fractale*), il existe plusieurs méthodes courantes de compression dont on va dire quelques mots.

a - RLE *(run length encoding)* ou **packbit.** Utilisée surtout dans le monde Mac, cette méthode repère les suites de caractères identiques : si le caractère *c* se répète *n* fois de suite, on écrit dans le fichier comprimé le caractère *n*, puis le caractère *c* au lieu des *n* caractères *c*. Ce type de compression est simple mais peu efficace (sauf dans les dessins au trait).

b - Huffman (ou *statistique*). On analyse le fichier pour y repérer les suites de bits se répètant souvent (étude statistique des fréquences d'apparition). Ces séquences seront inscrites dans une table (un dictionnaire) en face d'un code. Les suites les plus fréquentes auront des codes courts (2 ou 4 bits). On remplace dans le fichier comprimé la suite par son code et on joint la table au fichier créé. Ce type de compression est de règle dans la communication par fax (normes CCITT3 et 4), mais, dans ce cas, la table est fixée une fois pour toutes ; le récepteur la connaît, elle n'est donc pas jointe au fax. Le taux de compression atteint 5 à 8 pour le CCITT3 et 15 pour le CCITT4.

c - LZW (ou *codage adaptatif*). Là aussi, présence d'un dictionnaire, mais il est construit au fur et à mesure de la lecture du fichier à l'apparition de chaque séquence de caractères. Il n'y a pas d'étude statistique, le codage est progressif. Les séquences longues utilisent le code des séquences plus courtes rencontrées auparavant. L'algorithme est tel que le dictionnaire n'a pas besoin d'être joint au fichier : le décodeur le reconstruit au fur et à mesure de le lecture.

C'est le principe utilisé par de nombreux logiciels pour créer les fichiers ***.Z**, ***.zip**, ***.gif**, ***.arc**, ***.lha**, ***.arj**. La plupart de ces compresseurs assurent également l'*archivage*, c'est-à-dire le compactage de plusieurs fichiers en un seul. Sous Unix, on dissocie les deux fonctions : **uncompress** décomprime les fichiers de suffixe **.tar.Z** et **tar** les désarchive

Les algorithmes de compression LZW ont été brevetés et ont fait l'objet de plaintes devant les tribunaux. Le format ***.gif** en particulier ne serait pas du domaine public, pas plus que **uncompress**. La situation juridique n'est pas claire. Mais on peut utiliser sans crainte **gzip** (de Jean-Loup Gailly) et **ungzip** (de Mark Adler) pour les fichiers de type LZ et LZW, car ces outils échappent aux brevets (voir www.gzip.org). Quant au format **gif**, il pourrait à l'avenir être remplacé par le format **png**.

d - JPEG *(joint photographic experts group)*. Cette méthode est conçue pour comprimer les images à tons continus, i.e. de type photo noir-et-blanc ou couleurs ; elle autorise 16 millions de couleurs (alors que le format GIF n'en permet que 256). Elle comprime très mal textes, dessins au trait, schémas ... La méthode employée est complexe. Tout d'abord on analyse par blocs rectangulaires de 64 pixels la fréquence de variation de la couleur et on remplace l'image par son spectre de Fourier numérisé. Cette numérisation se fait à l'aide d'entiers obtenus par arrondi de la division avec un nombre arbitraire du terme spectral calculé. Les termes de faible amplitude vont devenir nuls et ne seront pas codés. Les fréquences élevées seront éliminées, leur amplitude étant faible, et l'image perdra un peu en définition. Le diviseur étant arbitraire, on peut régler le seuil des fréquences supprimées. Les images JPEG sont parfois transmises fréquence après fréquence et on les voit se construire à l'écran avec une résolution croissante.

Le deuxième mécanisme compresseur provient de la remarque que l'œil est plus sensible aux variations de luminance (éclairement moyen) que de chrominance (teinte). Ce constat avait déjà servi à réduire la bande passante dans tous les systèmes de télévision. Au lieu de décomposer la couleur d'un pixel en trois composantes RVB, on le fait en YC_rC_b (luminance, chrominance rouge, chrominance bleue ; ceci permet aux récepteurs en noir-et-blanc de recevoir une image correcte avec la seule information de luminance). En télévision, la bande passante accordée aux chrominances est la moitié de celles de la luminance ; les détails fins sur ses composantes sont perdus ; mais l'œil y est peu sensible, comme on l'a dit plus haut. Le codage JPEG procède de même. Si, pour coder un rectangle de 4 pixels, on utilise bien 4 octets (un par pixel) pour la luminance, on ne consacrera qu'un octet à chacune des chrominances de l'ensemble des 4 pixels. Au lieu des 12 octets (3×4) nécessaires en RVB, on n'en utilise que 6 ($4+2\times1$) dans le système YC_bC_r.

Enfin, les amplitudes du spectre de fréquence tronqué sont comprimées avec le procédé Huffman. C'est donc un procédé composite qui atteint aisément des taux de compression de 20 à 30. Les fichiers utilisant la norme JPEG sont de type **JFIF** (très simple) ou **SPIFF** (plus complexe, il permet une correction de type γ). Leurs descripteurs sont placés en tête et les nombres sont en format fixe. Signalons qu'il existe le format **MJPEG**, reposant sur les mêmes principes, pour comprimer les images de type cinéma.

Certains formats comprimés sont échangeables entre SE différents, spécialement ceux contenant des images, fichiers *.GIF, *.JPG .... Toutefois, codeur et décodeur sont propres à la machine qui les utilise.

3 - Compatibilité numérique

Le fichier peut comporter des **nombres**, entiers ou réels. Ils sont parfois exprimés en caractères ASCII, mais ce format étant très pénalisant en temps et en volume occupé, ils sont souvent codés en *binaire*. Le codage des réels est très complexe et relève de notre section 5 *(codes)*. Quant aux entiers, il faut savoir qu'il y a incompatibilité entre ceux des machines à composants Motorola (donc Macintosh) et ceux des machines à puces Intel (IBM-PC, *x*86, pentium, etc). Un entier court est formé de deux octets successifs écrits dans un ordre différent chez ces deux constructeurs (poids-faible-poids-fort pour Intel, l'inverse pour Motorola). Donc, un fichier comportant des entiers ne peut pas facilement être exporté d'un monde à l'autre. C'est le cas des fichiers-image : en effet, bien que les pixels de l'image soient codés par un seul caractère (3 si l'image est en couleurs), l'image est toujours accompagnée d'une partie descriptive donnant le nombre de lignes, de pixels par ligne ... On y trouve donc des entiers. Comme on l'a dit, certains formats assurent la *portabilité*, comme GIF, JPG ou TIFF. Les deux premiers imposent un format d'entier, le dernier utilise soit l'un, soit l'autre, en signalant son choix.

4 - Compatibilité de la table des caractères

Comme on l'a déjà dit, caractère et octet (8 bits) sont des termes équivalents. Or il n'existe de norme unique de correspondance octet-caractère que pour les 128 premières valeurs de l'octet (7 bits). Les *valeurs hautes* (128...255) sont codées de façon différente selon les systèmes d'exploitation, selon les logiciels ou même selon leur version... La compatibilité est donc réduite. Cette gêne provient de ce que l'écriture de l'anglais n'exige que 128 caractères. Autre conséquence, à l'origine les serveurs de réseaux ne transmettaient que 7 bits ; tout caractère de valeur *haute* était déformé (il existe encore en 1998 des serveurs Internet ne transmettant que 7 bits). Une solution à ce problème consiste à ne pas écrire dans le fichier de valeurs hautes pour l'octet. C'est une restriction bien gênante, ces valeurs hautes étant depuis longtemps utilisées pour coder soit des caractères spéciaux (cf. page 44), soit les lettres accentuées.

On peut décider de ramener tout caractère 8 bits à une suite de deux caractères 7 bits. Le TTX **T$_E$X** le fait. C'est également la solution adoptée par la plupart des logiciels de **messagerie**. La table de transformation peut être soit incorporée au fichier, soit faire l'objet d'une norme (*Mime, binhex* ...). Il en va de même pour le langage **HTML**, celui des fichiers accessibles sur Internet par les logiciels de navigation. Quand on utilise des caractères 8 bits, on peut également joindre au document leur table de définition. C'est ce que fait le langage **PostScript**, qui bénéficie ainsi d'une grande universalité.

Néanmoins, on assiste à une tendance vers l'unification en ce domaine. La table de codage 8 bits ISO-8859-1 ou **ISOLatin1** rallie de nombreux suffrages, du moins dans la francophonie : Unix et Mac l'emploient largement. Windows francisé utilise une table voisine en comblant des positions vides par des caractères supplémentaires comme les fameux œ (156) et Œ (140), oubliés par l'ISOLatin. Une autre table, *Unicode*, gourmande en temps et mémoire avec deux octets par caractère, mais acceptant tous les caractères mondiaux, pourrait elle aussi s'imposer dans certains cas, spécialement sur Internet.

Lorsque la table de codage du document est acceptée par le récepteur, le fichier est échangeable *au niveau du caractère*. Sinon, il faut procéder à une conversion, en général peu difficile. Certains logiciels assurent cette conversion. On peut également écrire un petit programme qui s'en chargera. Si on le fait peu souvent, c'est parce que l'incompatibilité de la table de caractères est rarement la seule en cause.

5 - Compatibilité logicielle

Pour un **logiciel simple**, comme un *éditeur*, désirant échanger un fichier simple (fichier de caractères, souvent appelé fichier ASCII), les exigences sur la compatibilité s'arrêtent à la table de caractères et ce genre d'échange réussit très souvent.

Mais en général, la compatibilité des caractères ne suffit pas. Les **logiciels élaborés** écrivent des codes ou des commandes dans leurs fichiers (par exemple pour indiquer une mise en forme, un changement de police, de *graisse* ...). Ces codes sont propres à chaque logiciel. La compatibilité entre versions différentes d'un même logiciel n'est pas du tout assurée. En général, pour surmonter cette difficulté, il faut demander la production (la *sortie*) d'un fichier dans un format conçu pour l'échange, comme ceux dont on parlera dans la prochaine section.

Le comble de la non-interchangeabilité est atteint avec les fichiers exécutables, formés presque entièrement de codes logiciels. Tout au plus, peut-on espérer qu'ils fonctionnent sous une version postérieure du système d'exploitation pour lequel ils ont été écrits.

6 - Formats d'échange

Il existe des formats de fichiers conçus spécialement pour l'échange, une fois franchie la barrière de compatibilité du disque. Ils sont donc bien adaptés aux échanges par réseaux. Ils résolvent le problème du codage des caractères comme on l'a expliqué dans la section 4. Quant à la barrière logicielle, elle est surmontée par l'emploi de codes de commande *littéraux* : ce sont des **mots** composés de plusieurs caractères de 7 bits et non plus des symboles de 8 bits.

Les fichiers issus d'un TTX (traitement de texte) peuvent théoriquement s'échanger sous le format **RTF** *(rich text format)*. Malheureusement, cette norme est souvent mal respectée. On a pu constater que les échanges entre mondes Mac et PC sont relativement satisfaisants, mais, curieusement, très difficiles entre Dos et Windows. Le format RTF préserve les caractères ainsi que la plus grande partie de la mise en forme du texte (hormis peut-être les *titres courants* et la position des *notes de bas de page*). Malgré tout, si le producteur du logiciel a respecté les normes, un fichier RTF est un excellent moyen d'échange.

On peut parfois demander la sortie d'un document TTX sous le format **HTML**, qui est, lui, tout à fait *portable*, mais ne préserve qu'une faible partie des enrichissements du texte. Signalons qu'un consortium international (dont fait partie l'INRIA) étudie actuellement un nouveau langage de ce type qui assurerait un maximum d'enrichissements. Un autre consortium tente de définir un langage d'échange rationnel pour les TTX (projet **TEI**, *text encoding initiative*) dans la lignée RTF, avec en plus des possibilités de marquage et des moyens facilitant la recherche par mots.

Les fichiers de format **PostScript** sont importables par n'importe quelle machine ; beaucoup de logiciels (dont les TTX) peuvent écrire des fichiers dans ce format. Il préserve la totalité des enrichissements, images comprises. Mais il ne permet qu'un échange *à sens unique*. Le fichier sera lu ou imprimé correctement, mais difficile à *reprendre*, à modifier. A notre connaissance, aucun TTX n'est capable d'*importer* du PostScript et de le convertir dans son propre langage. On ne peut modifier ce genre de fichier qu'en récrivant certaines de ses instructions (c'est difficile, car il s'agit d'un véritable langage de programmation).

La situation est encore plus tranchée pour le format **PDF** *(portable document file)*. Dérivé de PostScript, ce format universel permet de transcrire tous les raffinements des TTX, des tableurs, des images, etc. Il peut être affiché (avec effet de loupe facultatif) et imprimé sur n'importe quelle machine grâce au logiciel de lecture approprié, mais toujours gratuit (Acroread, Acrobat reader). Ce langage a été conçu pour ne pas permettre la modification du document et préserver ainsi le droit intellectuel de l'auteur. On peut toutefois ajouter au fichier des notes et des *références* (des renvois) avec le logiciel Exchange de la *suite* Acrobat commercialisée par Adobe. Il est également possible de le transformer en fichier PostScript, d'en reprogrammer le contenu et de le reconvertir en fichier PDF grâce au logiciel Distiller d'Adobe.

En ce qui concerne les **images**, le format **TIFF** a été conçu comme universel (une fois franchie la *barrière de la FAT*). Il est bien plus facile de coder une image dans le format TIFF (c'est pourquoi les scanners proposent toujours ce format) que de mettre au point un décodeur TIFF universel, parce que c'est un langage très souple avec beaucoup de possibilités (les champs de description de l'image sont placés n'importe où et peuvent décrire l'image de multiples façons). Un fichier TIFF peut être non comprimé ou comprimé sous diverses normes : packbit, Huffman, LZW, bientôt JPEG ... Les deux premiers caractères d'un fichier TIFF (MM ou II) indiquent sa provenance (Motorola ou Intel) ; le logiciel décodeur doit en tenir compte. TIFF est un très bon format de fichier image, mais on peut s'attendre à des déboires avec les logiciels interpréteurs insuffisamment universels.

Le format **JFIF** est beaucoup plus rigide : il impose son type de codage des entiers, les champs de descripteurs sont en tête du fichier et il utilise toujours la compression **JPEG** (on confond d'ailleurs souvent JPEG et JFIF). Malgré cette rigidité, il permet des échanges généralement réussis. On rappelle que l'image contenue est alors normalement une photo. Le format **SPIF**, voisin mais plus récent, devrait supplanter le format JFIF, avec des possibilités supplémentaires dont celle d'une correction γ (la compatibilité devrait être totale entre fichiers JFIF et SPIF). Le format **GIF** est également facilement échangeable. Cependant, sa "protection" par des brevets en limite beaucoup l'intérêt (voir fin §2c). A l'avenir, le format **PNG** pourrait le remplacer.

Pour changer le format d'une image pixellisée, on peut utiliser des logiciels gratuits, comme **IrfanView**. Si on désire pixelliser une image vectorielle, on peut le faire directement avec **Ghostview** ou bien la représenter à l'écran avec Ghostscript-Ghostview, la copier dans le presse-papier, supprimer les bords indésirables avec un logiciel de dessin comme MSPaint, puis la faire comprimer par IrfanView.

Un mot concernant les **sources** de programmes exécutables. Elles sont totalement échangeables sous deux conditions : le programme ne doit pas utiliser des fonctions réservées à un SE particulier (on peut parfois les transcrire) et la table de caractères pour les entrées-sorties doit être compatible ou modifiée.