# 1. Codage de l'information

**Exemple introductif**


Vous êtes en train de saisir un rapport sur un logiciel quelconque de traitement de texte. Lorsque vous appuyez sur la touche 'A' de votre clavier, votre action est repérée et traitée par le système d'exploitation ; le résultat est envoyé au programme:

1- Une touche du clavier a été actionnée
2- Cette touche est celle du 'A'
3- Le 'A' est envoyé par le système au traitement de texte
4- Le traitement de texte demande au système d'afficher cette lettre à l'écran
5- Le système affiche la lettre 'a' à l'écran

Ces cinq premières opérations se répètent tant que vous tapez du texte au clavier. Une fois votre texte terminé, vous désirez l'enregistrer sur une disquette. A l'aide du menu proposé par le logiciel, vous lancez cette opération de sauvegarde.

6- le logiciel demande au système d'exploitation de stocker votre rapport sur la disquette
7- le système vérifie la présence d'une disquette dans le lecteur
8- le système vérifie que la disquette dispose d'assez de place pour contenir votre rapport
9- le système réalise l'enregistrement

Sachant que toute information informatique est obligatoirement représentée par des suites de 1 et de 0 (numériques) : Comment le caractère 'A' est-il codé pour être transporté puis conservé en mémoire centrale et enfin affiché à l'écran ? Comment votre rapport est-il décrit sur la disquette, quel est son format ? Ces différentes questions concernent le codage de l'information.

## 1.1. Principe

Coder une information signifie la décrire selon un code numérique adopté **par convention**. Ainsi, si pour une application quelconque, les concepteurs décident qu'au mot *télévision* sera attaché le code 12, le programme manipulant ce mot saura, d'une part, le coder (*télévision* → 12) et, d'autre part, le décoder (12 → *télévision*). Pour être exprimés sous forme de 1 et de O, ces codes numériques sont ensuite simplement convertis en base 2.

**Rappel** : *de la base 10 à la base 2*

La base 10 (notre base décimale) utilise 10 symboles pour coder les quantités : les chiffres de 0 à 9. La base, quant à elle, n'utilise que 2 symboles : le 0 et le 1.

Pour obtenir l'équivalent d'un nombre écrit en base 10 en base 2 (ou X), il suffit de le diviser par 2 (ou par X si l'on veut le convertir en base X), puis de rediviser par 2 (ou X) à chaque fois le résultat obtenu par 2 (ou X) jusqu'à obtenir 0. L'écriture du nombre en base 2 (ou X) s'obtient en juxtaposant tous les restes obtenus dans les divisions consécutives.

Exemple : écrire 21 en base 2

| | | | | | | |
|---|---|---|---|---|---|---|
| 21 | divisé par | 2 | donne : | 10 | et il reste | **1** |
| 10 | divisé par | 2 | donne : | 5 | et il reste | **0** |
| 5 | divise par | 2 | donne : | 2 | et il reste | **1** |
| 2 | divisé par | 2 | donne : | 1 | et il reste | **0** |
| 1 | divisé par | 2 | donne : | 0 | et il reste | **1** |

Résultat : 21 s'écrit **1010** en base 2

**Rappel** : *de la base 2 à la base 10*

A l'inverse, pour retrouver l'écriture en base 10 d'un nombre écrit en base 2 (ou X), il suffit d'ajouter les puissances de 2 (ou X) de chaque chiffre composant le nombre.

Exemple écrire 1010 en base 10

$(\mathbf{1}x2^{4)} + (\mathbf{0}x2^{3)} + (\mathbf{1}x2^{2)} + (\mathbf{0}x2^{1)} + (\mathbf{1}x2^{0)} = (1x16) + 0 + (1x4) + 0 + (1x1) = \mathbf{21}$

L'unité d'information élémentaire en informatique est le **bit**. Le bit est un **emplacement mémoire**[1] pouvant prendre la valeur O ou 1. Sur un bit, nous pouvons donc décrire par convention deux informations différentes. Par exemple, dans une application de coloriage, au 1 pourrait correspondre le noir et au 0 le blanc alors que dans une application de cuisine le 1 pourrait représenter le sel et le 0 le poivre.
Bien évidemment, la plupart des applications manipulant bien plus de deux informations différentes, un seul bit ne suffit pas à toutes les décrire.
Prenons le cas de l'alphabet (lettres minuscules de a à z). Nous avons ici 26 informations différentes à décrire. Pour coder cet alphabet, nous considérons non pas un bit unique (codage de 2 valeurs distinctes) ni même deux bits (4 valeurs distinctes : 00, 01, 10 et 11) mais une série de cinq bits consécutifs. Nous obtenons avec cette série 32 valeurs différentes : 00000, 00001, 00010,…., 11110, 11111. Ces 32 valeurs nous permettent cette fois de décrire notre alphabet en associant, par exemple, le 'a' au code '00000', le 'b' au code '00001', etc.

**Conséquence**
Le nombre d'informations différentes que l'on peut décrire dépend du nombre de bits que l'on considère :

| | | | |
|---|---|---|---|
| 1 bit | code | $2^1 = 2$ informations différentes | 1, O |
| 2 bits | codent | $2^2 = 4$ informations différentes | 11, 10, 01, 00 |
| … | | | |
| 8 bits (1 octet) | codent | $2^8 = 256$ informations différentes | de 11111111 à 00000000 |
| … | | | |

**Le nombre d'informations différentes que l'on peut coder avec n bits est donc = $2^n$**

Les informations de base actuellement traitées en informatique sont les numériques, le texte, l'image (animée ou non), le son, la vidéo et, dans une moindre mesure, le « goût » (uniquement en reconnaissance) et « l'odeur » (uniquement en production). A partir de ces informations basiques, l'informatique peut décrire des informations plus qui sont une composition d'informations de base. Par exemple, la phrase « **Informatique** rime *avec* **numérique** » comporte du texte (information de base) et de la mise en forme (gras, italique, souligné) qui est une information graphique.

Parmi les informations numérisables (ou informatisables), nous distinguons :
- les informations de base
  - numériques
  - caractères
  - logiques
- les informations spécialisées
  - image
  - son
  - vidéo (que nous ne traiterons pas dans ce cours)
- les informations structurées

## 1.2. Informations de base

Par informations de base, on entend les informations auxquelles on peut directement rattacher un code numérique. Elles comprennent les numériques eux-mêmes, les caractères et les logiques.

[1] - Le terme « emplacement mémoire » ici est à prendre au sens large. Il désigne aussi bien un emplacement en mémoire centrale de l'ordinateur qu'un emplacement sur un support de stockage quelconque (clé USB, disuqe dur, cédérom, etc.).

### 1.2.1. Codage des numériques

Le codage des numériques peut sembler en soi relativement simple. En effet, pourquoi ne pas simplement utiliser leur correspondant en base 2. Par exemple, le nombre 10 s'écrirait 1010, 123 deviendrait 11110011, etc. Plusieurs problèmes s'opposent à cette solution.
Premièrement, les nombres sont infinis. Il faudrait donc pour les codes une infinité de bits ou d'octets cependant, aucune machine, aussi puissante soit-elle, ne dispose d'une telle capacité de mémoire. Deuxièmement, pour être comprise par un programme ou le processeur, une information doit posséder une longueur (en bits ou en octets) prédéfinie et fixe. Dans l'exemple précédent, 10 requiert 4 bits alors que 123 en demande 8. Enfin, les nombres décimaux et réels sont également des numériques et leur longueur peut être infinie ($\pi$ = 3,141592…). La simple réécriture en base 2 ne convient donc pas à la description des numériques.

En fait, il n'y a pas de solution unique au codage des numériques mais différentes approches selon le matériel et les logiciels. L'élément commun à ces approches est qu'aucun système ne peut gérer directement l'infini. Comme sur une simple calculatrice, il existe donc une taille maximale des nombres manipulés par chaque système.
Partant de ce constat, diverses solutions sont possibles :

- entiers entre 0 et 255 codés sur 1 octet (256 valeurs différentes)
- entiers relatifs entre -127 et +128 codés sur 1 octet
- entiers entre 0 et 65.535 codés sur 2 octets
- …

Tout est question de convention pour les programmes et le système manipulant ces valeurs.

### 1.2.2. Codage des logiques

En comparaison des numériques, le codage des logiques (booléens) est extrêmement simple. Ne comportant que deux valeurs (le vrai et le faux), il est en effet trivial de les coder.
Le vrai (*true* en anglais) peut prendre la valeur 1 et le faux (*false* en anglais) la valeur 0. A priori, 1 seul bit suffit donc à leur codage. Cependant, comme la plupart des informations traitées informatiquement nécessitent au minimum 1 octet pour leur codage, l'unité minimale de codage des informations est progressivement devenue l'octet. Les booléens seront donc codés par convention sur 1 octet :

- vrai : 11111111
- faux : 00000000

### 1.2.3. Codage des caractères

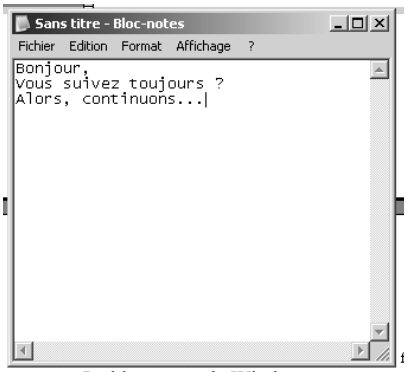
Le terme caractère est compris en informatique élément ayant un effet lors de son impression ou de son affichage. Les majuscules, les minuscules, les chiffres, les signes de ponctuation… sont donc des caractères au même titre que l'espace (le blanc), l'effacement arrière (suppression du caractère situé à gauche), la tabulation ou le retour chariot (passage à la ligne). Pour le français, on obtient donc l'ensemble suivant {a, b, c…z, A, B, C…Z, 0, 1…9, é, è, ê, ù, à…, espace, !, ", #, {, [, (, @...}.

Les américains ont rapidement proposé un codage des caractères anglo-saxons devenu rapidement un code international : le code ASCII (*American Standard Code for Information Interchange*). Dans ce code, les caractères sont décrits sut 1 octet ; 256 caractères distincts sont donc codifiables.

| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NUL (null) | 32 | 20 | 040 |   | Space | 64 | 40 | 100 | @ | @ | 96 | 60 | 140 | ` | ` |
| 1 | 1 | 001 | SOH (start of heading) | 33 | 21 | 041 | ! | ! | 65 | 41 | 101 | A | A | 97 | 61 | 141 | a | a |
| 2 | 2 | 002 | STX (start of text) | 34 | 22 | 042 | " | " | 66 | 42 | 102 | B | B | 98 | 62 | 142 | b | b |
| 3 | 3 | 003 | ETX (end of text) | 35 | 23 | 043 | # | # | 67 | 43 | 103 | C | C | 99 | 63 | 143 | c | c |
| 4 | 4 | 004 | EOT (end of transmission) | 36 | 24 | 044 | $ | $ | 68 | 44 | 104 | D | D | 100 | 64 | 144 | d | d |
| 5 | 5 | 005 | ENQ (enquiry) | 37 | 25 | 045 | % | % | 69 | 45 | 105 | E | E | 101 | 65 | 145 | e | e |
| 6 | 6 | 006 | ACK (acknowledge) | 38 | 26 | 046 | & | & | 70 | 46 | 106 | F | F | 102 | 66 | 146 | f | f |
| 7 | 7 | 007 | BEL (bell) | 39 | 27 | 047 | ' | ' | 71 | 47 | 107 | G | G | 103 | 67 | 147 | g | g |
| 8 | 8 | 010 | BS (backspace) | 40 | 28 | 050 | ( | ( | 72 | 48 | 110 | H | H | 104 | 68 | 150 | h | h |
| 9 | 9 | 011 | TAB (horizontal tab) | 41 | 29 | 051 | ) | ) | 73 | 49 | 111 | I | I | 105 | 69 | 151 | i | i |
| 10 | A | 012 | LF (NL line feed, new line) | 42 | 2A | 052 | * | * | 74 | 4A | 112 | J | J | 106 | 6A | 152 | j | j |
| 11 | B | 013 | VT (vertical tab) | 43 | 2B | 053 | + | + | 75 | 4B | 113 | K | K | 107 | 6B | 153 | k | k |
| 12 | C | 014 | FF (NP form feed, new page) | 44 | 2C | 054 | , | , | 76 | 4C | 114 | L | L | 108 | 6C | 154 | l | l |
| 13 | D | 015 | CR (carriage return) | 45 | 2D | 055 | - | - | 77 | 4D | 115 | M | M | 109 | 6D | 155 | m | m |
| 14 | E | 016 | SO (shift out) | 46 | 2E | 056 | . | . | 78 | 4E | 116 | N | N | 110 | 6E | 156 | n | n |
| 15 | F | 017 | SI (shift in) | 47 | 2F | 057 | / | / | 79 | 4F | 117 | O | O | 111 | 6F | 157 | o | o |
| 16 | 10 | 020 | DLE (data link escape) | 48 | 30 | 060 | 0 | 0 | 80 | 50 | 120 | P | P | 112 | 70 | 160 | p | p |
| 17 | 11 | 021 | DC1 (device control 1) | 49 | 31 | 061 | 1 | 1 | 81 | 51 | 121 | Q | Q | 113 | 71 | 161 | q | q |
| 18 | 12 | 022 | DC2 (device control 2) | 50 | 32 | 062 | 2 | 2 | 82 | 52 | 122 | R | R | 114 | 72 | 162 | r | r |
| 19 | 13 | 023 | DC3 (device control 3) | 51 | 33 | 063 | 3 | 3 | 83 | 53 | 123 | S | S | 115 | 73 | 163 | s | s |
| 20 | 14 | 024 | DC4 (device control 4) | 52 | 34 | 064 | 4 | 4 | 84 | 54 | 124 | T | T | 116 | 74 | 164 | t | t |
| 21 | 15 | 025 | NAK (negative acknowledge) | 53 | 35 | 065 | 5 | 5 | 85 | 55 | 125 | U | U | 117 | 75 | 165 | u | u |
| 22 | 16 | 026 | SYN (synchronous idle) | 54 | 36 | 066 | 6 | 6 | 86 | 56 | 126 | V | V | 118 | 76 | 166 | v | v |
| 23 | 17 | 027 | ETB (end of trans. block) | 55 | 37 | 067 | 7 | 7 | 87 | 57 | 127 | W | W | 119 | 77 | 167 | w | w |
| 24 | 18 | 030 | CAN (cancel) | 56 | 38 | 070 | 8 | 8 | 88 | 58 | 130 | X | X | 120 | 78 | 170 | x | x |
| 25 | 19 | 031 | EM (end of medium) | 57 | 39 | 071 | 9 | 9 | 89 | 59 | 131 | Y | Y | 121 | 79 | 171 | y | y |
| 26 | 1A | 032 | SUB (substitute) | 58 | 3A | 072 | : | : | 90 | 5A | 132 | Z | Z | 122 | 7A | 172 | z | z |
| 27 | 1B | 033 | ESC (escape) | 59 | 3B | 073 | ; | ; | 91 | 5B | 133 | [ | [ | 123 | 7B | 173 | { | { |
| 28 | 1C | 034 | FS (file separator) | 60 | 3C | 074 | < | < | 92 | 5C | 134 | \ | \ | 124 | 7C | 174 | | | \| |
| 29 | 1D | 035 | GS (group separator) | 61 | 3D | 075 | = | = | 93 | 5D | 135 | ] | ] | 125 | 7D | 175 | } | } |
| 30 | 1E | 036 | RS (record separator) | 62 | 3E | 076 | > | > | 94 | 5E | 136 | ^ | ^ | 126 | 7E | 176 | ~ | ~ |
| 31 | 1F | 037 | US (unit separator) | 63 | 3F | 077 | ? | ? | 95 | 5F | 137 | _ | _ | 127 | 7F | 177 | &#127; | DEL |

Dans ce code, au caractère 'A', par exemple, correspond la valeur 65 en décimal ou 10000001 en binaire. Américain d'origine, ce code ne comporte pas les caractères accentués. Une amélioration de l'ASCII, appelée « ASCII étendu », propose un codage (toujours sur 1 octet) pour tous les caractères de l'alphabet latin y compris les caractères accentués.

**Exemple** :
Le Bloc-notes est un petit éditeur de textes fourni avec le système Windows. Il permet la saisie et la manipulation de texte mais pas leur mise en forme (gras, italiques, centré…). Supposons que vous soyez en train de saisir un texte sur cet outil :



- Le bloc-notes de Windows -

Le format utilisé par cet outil pour sauvegarder les textes est appelé « texte brut ». En fait, dans le fichier, l'outil inscrit directement le code ASCII de chaque caractère composant le texte. Autrement dit, le format « texte brut » utilise un octet pour décrire la taille de chaque caractère.
Quelle est alors la taille de ce fichier ? Réfléchissez…comptez le nombre de caractères…B 1, o 2, n 3, j 4… attention, n'oubliez pas les espaces et les retours à la ligne….

Vous trouvez 54 caractères bravo ! La taille du fichier devrait être de 54 octets mais il y avait un piège ! Dans le code ASCII, le retour chariot compte en effet pour deux caractères : l'un pour indiquer que la ligne est finie, l'autre pour passer au début de la ligne suivante. Comme le texte comporte 2 retours chariot, la taille du fichier est donc de 56 octets.

Le développement international des communications informatisées (Internet) engendre que nous sommes de plus en plus amenés à recevoir des textes en provenance de tous les pays. Le code ASCII étant bien insuffisant pour traiter les alphabets latin, cyrillique, grecque… un nouveau standard de codage des caractères sur deux octets (65.535 caractères distincts) a fait son apparition : l'UNICODE.

**Conséquence**
La multiplicité des codages de caractères pose souvent problème aux utilisateurs. Les logiciels que l'on utilise ne comprennent souvent qu'un seul code. Il est donc fréquent d'ouvrir un texte quelconque et d'observer à l'écran des caractères étranges à la place, par exemple, d'un 'é' ou d'un 'ù'. Cela provient du fait que le texte initial a été codé selon la norme XXX et que votre logiciel l'interprète selon sa propre norme YYY.

## 1.3. Informations structurées

Par information structurée, nous entendons toute information pouvant être décrite à partir d'informations de base (numériques, booléens et caractères). Il n'existe pas de listes exhaustives de ce type d'information ni bien entendu de principes généraux pour leur codage. Nous ne traiterons ici que deux exemples classiques en informatique : le codage des nombres décimaux et celui de la mise en forme de textes.

### 1.3.1. Exemple : codage des nombres décimaux

Le problème consiste à définir un système de codage décrivant les nombres décimaux avec un nombre fixe d'octets. L'ensemble des nombres décimaux étant infini, la première restriction sera de limiter leur taille. En plus des chiffres composant le nombre, il nous faut coder le signe (+ ou -) et la place de la virgule.

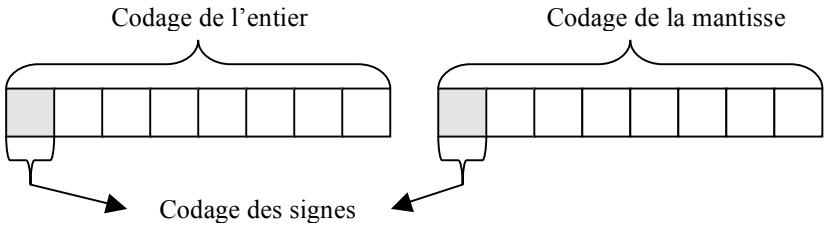
Bien que plusieurs approches soient possibles, nous ne présentons ici que la plus « classique ». Elle consiste à utiliser la notation dite scientifique qui consiste à décomposer le nombre en un entier et une puissance de 10 :

**1,45** va ainsi s'écrire **$145 \times 10^{-2}$** ($10^2$=100 alors $10^{-2}$ = 1/100 = 0,01)

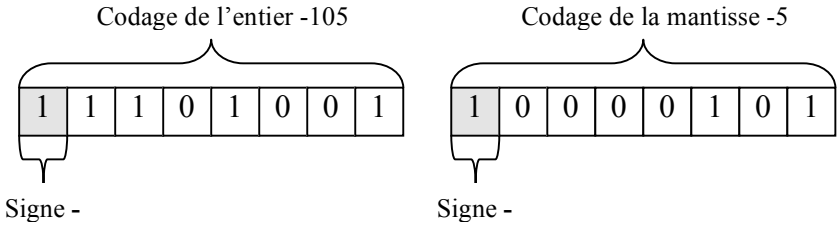
**-O,717** va lui s'écrire **$-717 \times 10^{3}$**

Décrire les décimaux revient donc à coder l'entier (145, -717…) avec son signe et la puissance de 10 (appelée mantisse) avec son signe (-2, 3…). Ne pouvant traiter l'infinité des décimaux, nous allons affecter 1 octet pour le codage de l'entier et 1 octet pour le codage de la puissance de 10. Les signes ne présentant que 2 valeurs distinctes, nous utiliserons le premier bit de chaque octet pour les coder (0 pour '+' et 1 pour '-'). Il restera donc 7 bits pour coder l'entier ou la mantisse soit $2^7$ = 128 possibilités de codage (nombre négatif ou positif entre 0 et 127). Pour étendre les possibilités, nous aurions pu affecter 2 octets au codage de l'entier : 7 bits + 8 bits = 15 bits soit 215 = 32.768 possibilités de codage (nombres entre 0 et 32.767).

**Structure du codage des décimaux**



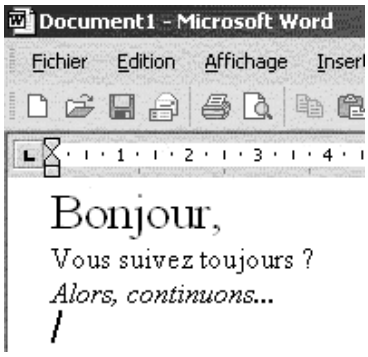

**Exemple : codage de -0,00105 = 105 x $10^{-5}$**




### 1.3.2. Exemple : codage des textes mis en forme

**Exemple** :
Reprenons l'exemple du §1.2.3. concernant le Bloc-notes de Windows. Sous le Bloc-notes, les textes saisis ne peuvent pas être mis en forme. Ce texte est maintenant passé sous le logiciel de traitement de textes Word (© Microsoft) pour être légèrement transformé :


- Le bloc-notes de Windows -




- Le logiciel Word -

Si le texte saisi reste bien le même, la mise en forme sous Word modifie son apparence : différentes tailles de caractères, les caractères de « Bonjour » sont en gras et ceux de « Alors, continuons… » en italiques. Pour le retrouver tel quel lors d'une future réutilisation, il est nécessaire de sauvegarder à la fois le texte et sa mise en forme. Quelles sont alors les techniques de codage possibles ?

Définir un format revient à analyser l'ensemble des informations à traiter et à leur associer une description en fonction de l'application.
Sur l'exemple précédent, quelles sont les informations décrivant le document ? Word est un logiciel de traitement de textes (= l'application) permettant donc la saisie et la mise en forme de textes en vue, généralement, de leur impression sur des feuilles de papier. De fait, la description d'un document saisi sous Word doit prendre en compte cet objectif. La taille de la feuille de papier, les marges, la position du texte dans la feuille, etc. sont autant d'informations à décrire en plus du texte et de sa mise en forme. Par contre, dans le cadre

d'une application où le document n'est destiné qu'à une lecture à l'écran (les pages Web, par exemple), la notion de feuille de papier (et donc de marges…) n'existe pas.

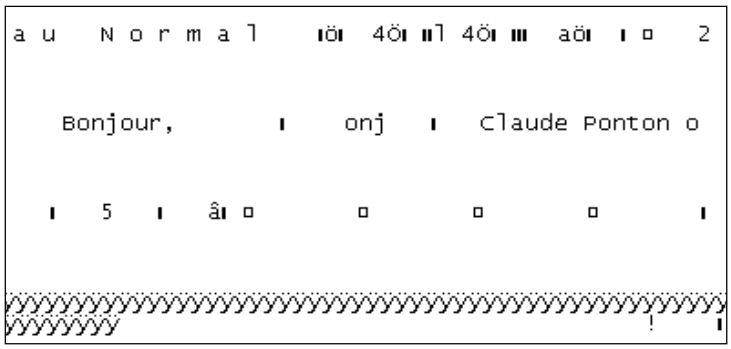
**Résumé**
Le choix d'un format de document dépend fortement de l'application et des données qu'elle manipule.

Les formats de description des documents de nature textuelle sont nombreux. Ils répondent aux objectifs internes de chacune des applications pour lesquelles ils ont été créés mais également à des objectifs de nature plus commerciale.

**Conséquence**
Le problème posé pour les utilisateurs est alors l'échange de documents. Imaginez que vous ayez saisi un rapport sur votre traitement XYZ qui le sauvegarde au format xyz. Vous donnez ce rapport sur disquette à un imprimeur pour qu'il vous en réalise un tirage. Malheureusement, l'imprimeur possède un traitement de texte ABC qui ne reconnaît que le format abc. Il lui est donc impossible d'ouvrir, et donc d'imprimer, votre document.

On distingue les formats « propriétaires » des formats « normalisés » et libres d'utilisation (RTF, HTML, XML…).

- Les formats propriétaires sont des formats déposés. Leur description interne est souvent tenue secrète et leur utilisation est soumise à un contrat entre la société dépositaire et les utilisateurs. Regardez, par exemple, le codage de notre exemple précédent au format Word (cf. ci-contre).
Difficile d'en comprendre le fonctionnement ! On retrouve bien des passages du texte d'origine (« Bonjour »), accompagné du nom de l'auteur (« Claude Ponton ») mais également un ensemble de codes totalement incompréhensibles.




**- Format Word -**

- La description interne des formats « normalisés » est accessible et reproductible. Leur utilisation est libre de droits. Dans le domaine du codage des documents textuels, le RTF (Rich Text Format) est certainement le plus répandu. Il offre une grande puissance de description des documents (mises en forme, auteur, graphiques, papier…) en utilisant uniquement des caractères ASCII (contrairement à Word qui utilise des codes propres à Microsoft) donc lisibles par quasiment tous les systèmes.

```
{\*\company DIP - Universit\'e9 Stendhal}{\notcharsws53}{\vern
\deftab708\widowctrl\ftnbj\aenddoc\hyphhotz425\noxlattoyen\exp
\jexpand\viewkind1\viewscale100\pgbrdrhead\pgbrdrfoot\splytwni
\fet0\sectd \linex0\headery708\footery708\colsx708\endnhere\se
\pndec\pnstart1\pnindent720\pnhang {\pntxta .}}{\*\pnseclvl4\p
{\*\pnseclvl7\pnlcrm\pnstart1\pnindent720\pnhang {\pntxtb ()}{\
\ql \li0\ri0\widctlpar\aspalpha\aspnum\faauto\adjustright\rin0
\par }{\insrsid2501953 Vous suivez toujours ?
\par }{\i\insrsid2501953\charrsid2501953 Alors, continuons...}
\par }{\i\insrsid2501953\charrsid2501953
\par }}
```

**- Format RTF -**