Jeux de caractères codés

Histoire

Structure

Jean-Marc Bourguet

Jeux de caractères codés

Histoire

Structure

jmblogo.pdf

Brouillon de la première édition 19 novembre 2010

Table des matières

Liste des tableaux

Ce qu'il y a de bien avec les normes,
c'est qu'il y en a tant parmis
lesquelles choisir. [1]

1 Introduction

Nous codons. Nous abstrayons. Nous codons les pensées que nous voulons communiquer sous forme de sons et nous appelons cela la parole. Nous abstrayons l'essentiel de la parole et nous appelons cela une langue. Nous codons les langues sous forme de dessins et nous appelons cela l'écriture. Nous abstrayons l'essentiel de ces dessins et nous appelons cela des lettres. Les codes sont utilisés partout dans notre vie. Les panneaux routiers en sont un. Les joueurs de bridge les utilisent pour donner des informations sur leur jeu à leur partenaire tout en essayant de les masquer à leurs adversaires.

Les codes permettent de donner un sens à des choses qui n'en n'ont pas par elles-mêmes. Ou d'en donner d'autres à ce qui en a déjà. Les codes évoluent et acquièrent des conventions supplémentaires qui permettent une expression au delà des limites de ce qu'ils étaient sensés coder au départ : un lecteur moderne est habitué à l'utilisation d'espaces pour séparer les mots, d'alinéas pour séparer les idées. Ces usages n'ont pas d'équivalent dans le language parlé et sont relativement récents comme le savent ceux qui ont dû déchiffrer des textes anciens.

À quoi servent donc tous ces codes ? Ils ont été introduits dans trois buts principaux.

Tout d'abord transmettre des informations — les exemples donnés relevaient tous de cet aspect ; conserver l'information n'est pas autre chose qu'un type de transmission où la distance temporelle prime sur la distance spaciale.

Une fois l'information codée, elle est accessible à tout ceux qui ont accès à la représentation codée et qui connaissent le code, ce qui peut être inconvéniant comme pour nos joueurs de bridge. Surtout que l'expérience a montré que la connaissance du code n'était pas toujours nécessaire pour extraire l'information parce qu'on pouvait souvent le déduire avec un effort plus ou moins important. Ce qui mène au deuxième objectif dans lequel on a conçu des codes : dissimuler l'information. La *cryptographie* cherche à dissimuler l'information à ceux qui n'ont pas la clé nécessaire, même s'ils en connaissent la présence et la nature du code utilisé, la *stéganographie* cherche à dissimuler la présence même de l'information — qui peut en plus être codée avec les techniques de la cryptographie.

Le troisième objectif est le traitement automatisé de l'information. Bien qu'historiquement il soit apparu en dernier lieu, sa présence est tellement présente dans nos vies, et encore plus pour l'audience de ce document, qu'il n'est pas besoin de s'étaller sur ce point.

Nous traiterons ici des codes utilisés pour la transmission et le traitement de l'information. Et plus précisément, ne seront traités que des codes basés sur l'écriture.

1. J'ai vu cet aphorisme attribué à Andrew Tanenbaum, mais je n'ai pas de référence.

2 Structure

Pour décrire les codes, il est utile d'avoir un modèle permettant d'en donner la structure. Il n'est pas de structure universellement adoptée, et pour compliquer les choses, le même vocabulaire est employé par différents auteurs dans des sens différents. La structure employée ici, inspirée de celles de [**?**], [**?**] et [**?**] mais qui n'est identique à aucune d'elles, est la suivante.

Le *caractère* est l'unité de base décrivant ce qui sera codé.

Un *jeu de caractères* ou encore *répertoire de caractères* est un ensemble de caractères, considéré complet pour l'usage auquel on destine le jeu de caractères.

Un *jeu de caractères codé*, c'est un répertoire de caractères auquel on ajoute une association d'un nombre à chaque caractère. Ce nombre est appelé *codet*. Il faut noter qu'un codet donné peut-être associé à plusieurs caractères.

Le *mécanisme de codage* permet de passer d'une suite de caractères, provenant d'un ou de plusieurs jeux de caractères codés, en une suite de *bytes*, l'unité de base de la représentation codée [2].

Il est possible de considérer que le problème du codage des caractères est terminé quand on a une suite de bytes. Mais il faut parfois tenir de problèmes pratiques qui existent chaque fois qu'on a à manipuler des données.

Premièrement, le byte du mécanisme de codage peut ne pas correspondre à l'unité de base du système cible. Il faut alors *sérialiser* la suite de bytes obtenus pour la transmettre ou la stocker.

Deuxièmement, même quand le byte du mécanisme de codage correspond à l'unité de base du système cible, certaines valeurs peuvent être en fait interdites. Ce problème existe principalement avec les protocoles de transmission. Il faut alors effectuer un *surcodage* pour respecter les contraintes du protocole utilisé.

Un *code* est donc l'association d'un mécanisme de codage, d'un ou plusieurs jeux de caratères codés et éventuellement de mécanismes de sérialisation et de surcodage.

2.1 Caractères

Les caractères sont naturellement les différents symboles utilisés par l'écriture : les lettres, les chiffres, les idéogrammes, les signes de ponctuation, ... Mais il ne faut pas confondre caractère et glyphe. Un *glyphe*, c'est un dessin. Le caractère, c'est la notion abstraite que certains glyphes désignent. « $\mathscr{A}$ », « A », « A » sont trois glyphes différents. Et ces trois glyphes sont des représentations du même caractère : la lettre A majuscule. Si ce principe est simple, son application est plus difficile. Examinons quelques problèmes potentiels.

Il n'est pas toujours simple de savoir si on a affaire à un caractère ou à plusieurs. Par exemple, est-ce que les accents doivent être considérés comme des caractères séparés des lettres ou est-ce que ce sont les lettres accentuées qui doivent être considérées

2. Le mot *byte* est aussi utilisé — mais pas dans ce texte — avec au moins deux autres acceptions : la plus petite unité adressable par un ordinateur et une chaîne d'exactement 8 bits. Dans ce dernier sens, il faut préférer le mot *octet* dont c'est l'unique signification.

comme des caractères ? Est-ce que « à » est un caractère ou le résultat de la combinaison d'un « a » avec un « ` » ? Pour une langue comme le français où les grammairiens considèrent les lettres accentuées comme des entités non décomposables et où les combinaisons possibles sont relativement peu nombreuses, avoir un caractère pour chaque lettre accentuées a des avantages. Mais quand le nombre de combinaisons possibles augmente, ou si la grammaire considère les accents comme des entités, il peut être intéressant de structurer plus le code et de considérer les accents comme des caractères en soi pouvant être combinés avec les lettres.

Les ligatures ont le même genre de problèmes. Les ligatures linguistiques comme « œ » sont fortement candidates à être des caractères, la substition de la paire de caractères « oe » par « œ » ne pouvant être faite automatiquement qu'en utilisant un dictionnaire ou une connaissance approfondie de la langue utilisée. Mais qu'en est-il des ligatures typographiques (« ﬁ » plutôt que « fi ») dont l'utilisation résulte plus de considérations typographiques que linguistiques ? Permettre d'effectuer des recherches sans devoir les spécifier est quasiment indispensable. Et leur introduction automatique par un moteur de rendu est certainement à envisager. Mais les considérations linguistiques ne sont pas totalement absentes de leur utilisation. Ainsi, même si on dispose de la ligature « ﬀ » on peut préférer en anglais « shelfful » (sans ligature) à « shelffful » (avec une ligature) pour éviter que la ligature ne combine des lettres de deux syllabes. En Allemand, ce n'est même plus une préférence, et l'usage est quasiment universel de ne pas faire de ligatures entre deux lettres d'un mot composé (donc « auflage » et pas « auflage »). Et si on écrit du Turc, qui dispose d'un « i » et d'un « ı », remplacer « fi » par « ﬁ » pourrait introduire des ambiguités. De même, si on veut effectuer une transcription précise de documents anciens faisant un usage abondant et non systématique de ligatures, leur introduction automatique est à éviter.

Parfois, ce qui est considéré comme une lettre par les locuteurs d'une langue correspond à plusieurs glyphes, le choix du glyphe se faisant en tenant compte du contexte. Par exemple la lettre grecque sigma a les variantes « ς » et « σ » qu'il faut utiliser suivant qu'il s'agit de la dernière lettre d'un mot ou pas. Est-ce qu'il faut avoir deux caractères, un pour chacune des deux variantes, ou bien un seul ? Et si on considère d'autres langues, le nombre de formes peut être beaucoup plus important : l'arabe par exemple a jusqu'à quatre formes pour ses lettres (version isolée, initiale, finale et en milieu de mot). Si avoir un caractère pour chaque forme peut être sensé pour le grec, multiplier par quatre le nombre de caractères est exclus quand on a un nombre de codets limités.

Si généralement plusieurs glyphes peuvent représenter un caractère, parfois, deux caractères distincts peuvent être représentés par un même glyphe. Par exemple, le « A » de tout à l'heure pourrait être aussi bien un alpha majuscule. Est-ce qu'il faut avoir un caractère pour la lettre A majuscule et un autre pour la lettre alpha majuscule ou bien n'avoir qu'un caractère ?

Ces questions ne sont pas simples. En fait, il n'y a pas de bonne réponse universelle à leur apporter. Suivant les contraintes et les applications prévues, on peut faire — et on a fait — des choix différents. Même quelque chose d'apparemment aussi simple que de dire que « $\mathscr{A}$ » et « A » sont des glyphes différents d'un même caractère peut être remis en question si on considère les mathématiques où ces deux glyphes peuvent désigner des choses différentes et donc leur différence ne pas simplement être un embellissement typographique.

En plus des caractères représentant les symboles de l'écriture, les jeux de caractères comportent souvent une série d'autres caractères sans correspondance graphique, souvent sans même de correspondance linguistique, qui ont été ajoutés parce qu'ils étaient utiles. Ce sont les *caractères de commande* (souvent appelés *caractères de contrôle*, expression plus proche de la formulation anglaise) car la plupart d'entre eux servent à commander les périphériques.

Entre les caractères de commande et les caractères graphiques, il y a l'espace, caractère dont on s'est demandé s'il est un caractère graphique complètement blanc ou un caractère de commande indiquant qu'il faut avancer la position d'écriture. La première alternative semble s'être imposée.

On peut diviser les caractères de commandes en plusieurs classes (voir [**?**]) :
– ceux servant à contrôler les transmissions ;
– ceux servant à contrôler la mise en page ;
– ceux servant à gérer les codes ;
– ceux servant à commander les périphériques.

Les fonctions que l'on désire encoder avec les caractères de contrôle sont tellement nombreuses qu'on se sert de combinaisons de caractères, commençant souvent par un caractère de commande nommé ESC (voir [**?**], [**?**]).

2.2 Jeu de caractères codé

Pour rappel un *jeu de caractères codé*, c'est un répertoire de caractères auquel on ajoute une association d'un nombre à chaque caractère. Ce nombre est appelé *codet*. On emploie aussi les termes de *charset*, *codeset*, *page de code* (quasiment exclusivement chez IBM et Microsoft) pour désigner les jeux de caractères codés. « Codet » n'est pas non plus dénué de synonymes : *valeur scalaire*, *élément de code*, *position de code*, *point de code* et souvent même *code* malgré l'ambiguïté possible.

Parfois, en particulier pour les charsets comportant un nombre important de caractères et conçus pour être utilisés avec un code respectant la structure d'ISO 2022 (voir **??**), les codets ne sont pas un nombre unique mais des n-uplets.

Si chaque caractère du répertoire doit avoir un codet, un codet donné peut correspondre à plusieurs caractères. Dans ce cas un mécanisme doit exister pour permettre de lever l'ambiguité, l'interprétation d'un codet dépend donc de ceux qui le précèdent. En pratique les jeux de caractères associant plusieurs caractères à un codet sont des jeux modaux : l'interprétation dépend d'un mode courant et seuls certains caractères de contrôle peuvent changer le mode courant.

2.3 Mécanisme de codage

Le mécanisme de codage décrit comment passer des codets à une suite de bytes. Le plus simple, c'est la fonction identité (chaque codet produit un byte de la valeur du codet) et il est souvent utilisé implicitement.

Certains mécanismes permettent de combiner des jeux de caractères codés différents. Dans ce cas, le mécanisme peut toujours être très simple : continuer à utiliser la

fonction identité tout en imposant que les codets soient différents, ou plus compliqué (voir **??**) en utilisant des caractères de contrôle pour indiquer le changement de charset.

Les jeux de caractères codés comportant un nombre important de caractères (ce qui est le cas de ceux permettant de coder les écritures idéographiques) ont un problème avec l'utilisation de la fonction identité : il faudrait alors des bytes de grande taille. Les mécanismes de codage qui sont destinés à être employés avec eux prévoient donc des moyens d'utiliser des bytes plus petits, mais en utilisant parfois plusieurs bytes pour un caractère.

Il faut noter que la différence entre charset et mécanisme de codage n'est pas toujours faite. C'est souvent le cas quand on envisage l'utilisation de la fonction identité comme mécanisme de codage, mais pas exclusivement. On parle parfois de charset multibyte pour désigner l'utilisation d'un charset ayant un nombre important de caractères et d'un mécanisme de codage utilisant parfois plus d'un byte pour représenter un caractère.

2.4 Sérialisation

Si la taille du byte ne correspond pas à une unité manipulée naturellement par le système, il faut alors prévoir comment représenter ceux-ci.

On peut avoir à regrouper plusieurs bytes en un mot. Par exemple, le PDP-10 a des mots de 36 bits. Il a été utilisé avec des mécanismes de codage utilisant des bytes de 6 bits (on mettait 6 caractères par mot), de 7 bits (on mettait 5 caractères par mot, et un bit était inutilisé) et de 9 bits (4 caractères par mot). Savoir que plusieurs bytes se trouvent dans le même mot ne suffit pas, il faut encore connaître leur position respective. Dans le cas du PDP-10, c'était déterminé par le jeu d'instruction de ce processeur qui a une notion de pointeur vers bytes où les pointeurs en question contiennent le nombre de bits du byte, de 1 à 36. Naturellement, la position des bytes dans le mot correspondait à ce qu'il fallait pour utiliser ces instructions.

On peut aussi avoir le problème inverse et devoir découper un byte en plusieurs unités si ce n'est pas prévu par le mécanisme de codage. Par exemple Unicode définit un mécanisme de codage appelé UTF-16 qui utilise des bytes de 16 bits. Pour les représenter sur un ordinateur ayant comme unité de base l'octet on a le choix : si on place la partie la plus significative du byte en premier, la sérialisation est *gros-boutiste*, si la partie la moins significative est en premier, elle est *petit-boutiste*.

2.5 Surcodage

Le surcodage est nécessaire quand le résultat, éventuellement sérialisé, du codage n'est pas utilisable comme tel. Par exemple parce qu'il faut le transmettre avec un protocole qui interdit certaines valeurs.

Le surcodage est généralement transparent pour l'utilisateur ; et le système qui l'effectue n'a souvent même pas connaissance qu'il est en train de manipuler des caractères plutôt qu'un autre type de donnée.

L'exemple le plus commun de surcodage visible — heureusement de moins en moins — est avec le protocole SMTP [**?**], utilisé pour transmettre les couriels. Il est définit comme opérant sur des octets, mais sans garantie de conservation du bit de poids fort. De plus les lignes sont de longueurs limitées. Ça fonctionne bien avec l'ASCII — qui est un code sur 7 bits — mais les données utilisant un code sur 8 bits peuvent être modifiées dans le transport.

Un protocole annexe, MIME [**?**], résouds ce problème — et d'autres, par exemple il permet d'indiquer dans le message le charset utilisé, ce qui est un autre problème du SMTP — tout en conservant une bonne compatibilité avec les systèmes antérieurs. Bonne compatibilité en ce sens qu'avec un logiciel qui ne comprend pas MIME (ce qui est maintenant rare), on se retrouve avec une partie du message dégradée, mais souvent encore compréhensible pour autant que le message soit écrit en utilisant un charset qui est une extension de l'ASCII.

MIME définit deux surcodages qui peuvent être utilisés pour transmettre un code sur 8 bits : un (*quoted printable*) à utiliser plutôt quand les données sont principalement des codets avec les 8^{e} bit à 0 car seuls les codets ayant le 8^{e} bit à 1 sont transformés ce qui laisse souvent le message surcodé compréhensible. Par exemple si un couriel est écrit en utilisant le code ISO 8859-1, les lettres sans accents sont transmises sans changement et un « é », par exemple, devient « =E9 » — le codet de « é » étant `E9` exprimé en hexadécimal. Le deuxième surcodage (*base64*) est à utiliser dans les autres cas, et le message est alors incompréhensible sans décodeur.

3 Quelques codes

Les codes utilisés en transmission ont une longue histoire. Au deuxième siècle avant notre ère, le général et historien grec Polybe (Πολύβιος, v. 200 – v. 120 av. J.-C.) propose un système pour coder l'alphabet grec en utilisant deux groupes de cinq torches. Ce système a eu un certain succès : au XX^{e} siècle les prisonniers de guerre américains au Viêt Nam auraient utilisé une variante de ce système, adapté à l'alphabet latin [**?**].

Les marins de la Grèce antique utilisaient déjà des drapeaux pour communiquer entre navires d'une même flotte. Ce moyen de communication était relativement inflexible mais en 1738, Bertrand François Mahé de la Bourdonnais a conçu un code plus complexe : dix drapeaux différents codaient les dix chiffres, l'utilisation de trois drapeaux permettaient de coder 1000 messages, un dictionnaire donnait la correspondance entre les nombres et les messages. Ce système ne fut pas directement employé, mais influença vraisemblablement Lord Richard Howe qui inclu un mécanisme similaire dans son « The Howe Code ». Par la suite, ce code fut complèté par Sir Home Popham dans son « Telegraphic Signals or Marine Vocabulary » qui finit de le perfectionner en codant individuellement les lettres, permettant ainsi d'épeller ce qui ne se trouve pas dans le dictionnaire. Ce qui, en 1805 à Trafalgar, permit à l'amiral Horatio Nelson de transmettre le célèbre message « England expects that every man will do his duty » bien que « duty » ne soit pas dans le dictionnaire ([**?**]).

À la fin du dix-huitième siècle, les frères Chappe établirent un système de sémaphores composés de tours équipées de bras articulés. Comme le code de Popham, le code utilisé était aussi un système à dictionnaire disposant d'un moyen d'épeller ce qui

n'était pas dans le dictionnaire. Ce système fut utilisé par Napoléon pour gouverner son empire et commander ses armées, lui donnant un avantage sur ses ennemis [**?**, **?**, **?**].

3.1 Le code Morse

A	·—	M	——	Y	—·——
B	—···	N	—·	Z	——··
C	—·—·	O	———	0	—————
D	—··	P	·——·	1	·————
E	·	Q	——·—	2	··———
F	··—·	R	·—·	3	···——
G	——·	S	···	4	····—
H	····	T	—	5	·····
I	··	U	··—	6	—····
J	·———	V	···—	7	——···
K	—·—	W	·——	8	———··
L	·—··	X	—··—	9	————·

TABLE 1: Le code Morse

En 1837 Samuel Morse brevette un télégraphe électrique. Le code qu'il propose alors d'utiliser avec son système est apparemment basé uniquement sur un dictionnaire, mais il n'a pas été utilisé. C'est son assistant, Alfred Vail, qui aurait travaillé sur le code et proposé par après ce qu'on appelle le code Morse (voir table **??**). Ce code est basé sur cinq symboles (deux durées de courant sur la ligne — représentées par « · » et « — » dans la table — et trois durées d'absence de courant — séparant les points et les traits, les lettres et les mots). Dans ce code tous les caractères codés ne sont pas composés du même nombre de symboles. Après tout, c'est un principe de compression bien connu que de coder avec moins de symboles ce qui est fréquent. C'est donc naturellement que le « E », lettre la plus fréquente en anglais comme elle l'est en français, est codée par « · ». Une telle compression est un avantage dans une application de transmission, mais est plutôt une nuisance quand il s'agit d'effectuer un traitement automatique.

3.2 Les codes Baudot, Murray et les alphabets télégraphiques internationnaux n° 1 et 2

Au début des années 1870[3] et toujours pour les transmissions télégraphiques, Émile Baudot invente des systèmes télégraphiques imprimant automatiquement les messages à leur réception. Pour permettre cette automatisation, il abandonne le code Morse, trop compliqué à décoder automatiquement (il faudra attendre le XX^{e}siècle pour avoir un système capable de le décoder et de l'imprimer).

3. Il optient un brevet le 17 juin 1874

			Lettres		Chiffres	
HEX			0	1	0	1
	BIN		0	1	0	1
		OCT	*0*	*2*	*0*	*2*
0	0000	*0*	0 (?)	16 (LS)	0 (?)	16 (LS)
1	0001	*1*	1 A	17 t	1 1	17 .
2	0010	*2*	2 E	18 X	2 2	18 ,
3	0011	*3*	3 É	19 Z	3 &	19 :
4	0100	*4*	4 Y	20 S	4 3	20 ;
5	0101	*5*	5 U	21 T	5 4	21 !
6	0110	*6*	6 I	22 W	6 o	22 ?
7	0111	*7*	7 O	23 V	7 5	23 ’
8	1000	*0*	8 (FS)	24 (ERA)	8 (FS)	24 (ERA)
9	1001	*1*	9 J	25 K	9 6	25 (
A	1010	*2*	10 G	26 M	10 7	26)
B	1011	*3*	11 H	27 L	11 h	27 =
C	1100	*4*	12 B	28 R	12 8	28 —
D	1101	*5*	13 C	29 Q	13 9	29 /
E	1110	*6*	14 F	30 N	14 f	30 N°
F	1111	*7*	15 D	31 P	15 0	31 %
		OCT	*1*	*3*	*1*	*3*

(?) pas utilisé
LS Letter-Blank (passage aux lettres + espace)
FS Figure-Blank (passage aux chiffres + espace)
ERA Erasure (annulation du caractère précédent)

TABLE 2: Le code Baudot

			Lettres		Chiffres	
HEX			0	1	0	1
	BIN		0	1	0	1
		OCT	*0*	*2*	*0*	*2*
0	0000	*0*	0 (1)	16 (LS)	0 (1)	16 (LS)
1	0001	*1*	1 A	17 (LF)	1 1	17 (LF)
2	0010	*2*	2 E	18 X	2 2	18 ,
3	0011	*3*	3 (CR)	19 Z	3 (CR)	19 :
4	0100	*4*	4 Y	20 S	4 3	20 .
5	0101	*5*	5 U	21 T	5 4	21 (2)
6	0110	*6*	6 I	22 W	6 (2)	22 ?
7	0111	*7*	7 O	23 V	7 5	23 ’
8	1000	*0*	8 (LS)	24 (ERA)	8 (FS)	24 (ERA)
9	1001	*1*	9 J	25 K	9 6	25 (
A	1010	*2*	10 G	26 M	10 7	26)
B	1011	*3*	11 H	27 L	11 +	27 =
C	1100	*4*	12 B	28 R	12 8	28 -
D	1101	*5*	13 C	29 Q	13 9	29 /
E	1110	*6*	14 F	30 N	14 (2)	30 (2)
F	1111	*7*	15 D	31 P	15 0	31 %
		OCT	*1*	*3*	*1*	*3*

(1) pas utilisé
(2) usage local (lettres accentuées par exemple)
CR Carriage-return (retour chariot)
LF Line-feed (passage à la ligne)
LS Letter-space (espace + passage aux lettres)
FS Figure-space (espace + passage aux chiffres)

TABLE 3: Le code CCITT#1 ou alphabet télégraphique international n° 1

			Lettres		Chiffres	
HEX			0	1	0	1
	BIN		0	1	0	1
		OCT	*0*	*2*	*0*	*2*
0	0000	*0*	0 (2)	16 E	0 (2)	16 3
1	0001	*1*	1 T	17 Z	1 5	17 +
2	0010	*2*	2 (CR)	18 D	2 (CR)	18 (WRU)
3	0011	*3*	3 O	19 B	3 9	19 ?
4	0100	*4*	4 (SP)	20 S	4 (SP)	20 ’
5	0101	*5*	5 H	21 Y	5 (1)	21 6
6	0110	*6*	6 N	22 F	6 ,	22 (1)
7	0111	*7*	7 M	23 X	7 .	23 /
8	1000	*0*	8 (LF)	24 A	8 (LF)	24 -
9	1001	*1*	9 L	25 W	9)	25 2
A	1010	*2*	10 R	26 J	10 4	26 (Bell)
B	1011	*3*	11 G	27 (FS)	11 (1)	27 (FS)
C	1100	*4*	12 I	28 U	12 8	28 7
D	1101	*5*	13 P	29 Q	13 0	29 1
E	1110	*6*	14 C	30 K	14 :	30 (
F	1111	*7*	15 V	31 (LS)	15 =	31 (LS)
		OCT	*1*	*3*	*1*	*3*

(1) usage local (lettres accentuées par exemple)
(2) pas utilisé
WRU Who are you
CR Carriage-return (retour chariot)
LF Line-feed (passage à la ligne)
LS Letter-shift (passage aux lettres)
FS Figure-shift (passage aux chiffres)
SP Space (espace)

TABLE 4: Le code CCITT#2 ou alphabet télégraphique international n° 2

Ce système employait un code (voir table **??**) à deux symboles (impulsion positive ou négative), c'est donc un code binaire, comme tout ceux que nous rencontrerons par la suite. Les caractères étaient codés en utilisant des motifs de cinq *bits*[4] (*moments* dans la terminologie de l'époque). Le code aurait été conçu de manière à limiter la fatigue de l'opérateur, structure que la représentation tabulaire utilisée ici ne met pas en valeur.

Ce code est un code modal. Pour permettre plus de 32 caractères (ce est nécessaire si on veut coder au moins 26 lettres plus 10 chiffres), il utilise deux modes : certains motifs sont utilisés pour coder deux caractères différents, le choix entre eux étant effectué en fonction du mode courant. Le changement de mode est notifié par l'envoi des caractères notés (FS) et (LS) dans le tableau.

Au départ, les codets sont à interprété suivant la partie « Lettres » du tableau jusqu'à réception d'un (FS) qui indique que les codets suivants sont à interpréter comme indiqué dans la partie « Chiffres ». Un (LS) permet de repasser aux lettres.

Le code ne prévoit pas de représentation pour l'espace : les caractères utilisés pour changer de modes, (FS) et (LS), ont aussi cette fonction. Ce qui présente un avantage : on est certain qui si un de ces caractères est corromptu – et donc que le changement de mode ne se fait pas –, il en viendra bientôt un espace qui remettra les choses en ordre. Une corruption a donc un effet localisé. Un autre avantage du système est qu'il ne consomme que deux codets pour les trois fonctions. Un inconvénient est que tout changement de mode s'accompagne d'un espace.

Naturellemet, quand le système de Baudot se répandit, certains codets furent utilisés pour d'autres usages sur certaines lignes.

Vers 1900, Donald Murray développa un système télégraphique inspiré de celui de Baudot, mais remplaça le clavier à cinq touches par un clavier semblable à celui d'une machine à écrire. Il changea aussi le code, cette fois-ci avec pour objectif de faciliter la mécanique de son système et d'en diminuer l'usure. En plus d'une assignation différentes des codets, le code de Murray séparait la fonction de changement de mode de celle d'espace, permettant de changer de mode pour des caractères consécutifs.

Et comme pour le code de Baudot, différentes versions se mirent rapidement à exister.

En plus de ces deux codes de base et de leurs variantes, il y eu d'autres systèmes télégraphiques utilisant d'autres codes encore. Ces divergences commencèrent à poser des problèmes, si bien que quand le Comité Consultatif International Téléphonique et Télégraphique fut formé dans la seconde moitié des années 20 parmis ses objectifs se trouvait la définition d'un code à cinq moments. Ce qui fut fait au début des années 30 fut la définition de deux alphabets télégraphiques internationnaux, un basé sur le code Baudot (table **??**) et un basé sur le code Murray (table **??**).

En 2009, ce dernier était toujours utilisé par certains systèmes de télécommunication [**?**].

À noter qu'il est d'usage d'appeler code « Baudot » les codes à cinq moments quel que soient leur rapport avec le code inventé par Baudot.

[**?**] décrit l'histoire des codes Baudot et Murray et de leur normalisation en tant qu'alphabets télégraphiques internationnaux. [**?**] est la réimpression de 1934 d'un manuel de 1919 destiné aux opératrices de télégraphes Baudot. [**?**] contient une table avec les caractères locaux alloués aux

4. Une première version a utilisé un code à 6 bits.

Royaumes-Unis, en France, en Allemagne, en Scandinavie ainsi que le code US-TTY, une variante du code de Murray utilisée aux États-Unis.

Suivant les sources, les motifs de bits sont parfois inversés. Ces codes sont définis en considérant leur ordre de transmission. La question est alors de savoir si le bit de poid faible est transmis en premier ou non. Les protocoles de transmission en série avec lesquels je suis famillier transmettant d'abord le bit de poid faible et en l'absence de source claire sur ce sujet j'ai suivi cette convention qui est aussi celle utilisée par [**?**].

3.3 BCDIC

Quittons le domaine de transmission pour regarder ce qui se passait pour l'automatisation du traitement des données. Le developpement de celui-ci est plus récent. C'est durant les années 1880 qu'Herman Hollerith développe les premières tabulatrices qui seront utilisées pour le traitement du recensement américain de 1890. Il fonde alors une compagnie qui deviendra après quelques fusions IBM. Dans ces systèmes, les informations sont codées par des trous dans des cartes. Les premiers codes pour les cartes perforées avaient 12 caractères : les 10 chiffres et deux caractères de contrôle. Les cartes étaient composées d'un certain nombre de colonnes, chaque colonne ayant 12 positions, une pour chaque caractère possible. Les rangées portaient comme nom le chiffre qu'elle codait. Le nom des deux rangées supplémentaires varie suivant les sources — 10 et 11, 11 et 12, A et B, X et Y — ce texte utilise X et Y. Une des utilisations des caractères de contrôle X et Y fut d'indiquer si le nombre était positif ou négatif.

Ce code fut étendu à 40 caractères dans les années 30 puis à 48 caractères dans les années 50. Les caractères correspondent alors à des combinaisons de plusieurs trous. Les trous sont séparés en deux groupes : ceux sur des rangées dites *de zones* et ceux sur des rangées *de chiffres* permettant d'adresser un tableau a double entrée — avec souvent quelques exceptions. Au départ les rangées de zones étaient les rangées X et Y, et les rangées de chiffres... celles des chiffres. Mais par la suite les rangées 0 et 9 ont été utilisées comme rangées de zones. Jusqu'alors, nous avons un code défini purement pour les cartes perforées. On pourrait le considérer comme un code utilisant un byte de 12 bits ; mais il n'était pas perçu comme tel.

Tout comme « code Baudot » désigne tout code à cinq moments, « code Hollerith » désigne tout code pour carte perforée utilisant douzes rangées (d'autres types de codes ont été utilisés avec les cartes perforées, certains plus proches d'un codage en binaire). [**?**] indique qu'en 1964 une enquête auprès de huit fabriquand en a dénombré 21 versions ayant en commun le code de 43 caractères (les chiffres, les lettres, l'espace et « . » , « * », « / », « - », « $ »).

Lorsqu'on s'est mis à utiliser des ordinateurs et non plus uniquement des tabulatrices, on a ajouté au code Hollerith, défini par des combinaison de trous dans les 12 rangées des cartes, un codage numérique interne, généralement sur 6 bits. Le code BCDIC *Binary Coded Decimal Interchange Code* (voir table **??**) est le nom de celui utilisé par IBM. Naturellement, il y a eu évolution et la version présentée ici est le résultat final de cette évolution. Une caractéristique importante de ce code est qu'il respecte la structure du code pour carte perforée : les colonnes correspondent à une combinaison donnée dans les rangées de zones et les rangées correspondent à une combinaison de

	HEX		0	1	2	3
	BIN		00	01	10	11
		OCT	*0*	*2*	*4*	*6*
0	0000	*0*	0 (SP)	16 ƀ	32 -	48 & ou +
1	0001	*1*	1 1	17 /	33 J	49 A
2	0010	*2*	2 2	18 S	34 K	50 B
3	0011	*3*	3 3	19 T	35 L	51 C
4	0100	*4*	4 4	20 U	36 M	52 D
5	0101	*5*	5 5	21 V	37 N	53 E
6	0110	*6*	6 6	22 W	38 O	54 F
7	0111	*7*	7 7	23 X	39 P	55 G
8	1000	*0*	8 8	24 Y	40 Q	56 H
9	1001	*1*	9 9	25 Z	41 R	57 I
A	1010	*2*	10 0	26 ǂ	42 !	58 ?
B	1011	*3*	11 # ou =	27 ,	43 $	59 .
C	1100	*4*	12 @ ou ’	28 % ou (	44 *	60 ¤ ou)
D	1101	*5*	13 :	29 γ	45]	61 [
E	1110	*6*	14 >	30 \	46 ;	62 <
F	1111	*7*	15 ✓	31 ǂǂ	47 Δ	63 ǂǂǂ
		OCT	*1*	*3*	*5*	*7*

SP	Space (espace)
ƀ	Substitute Blank
Δ	Mode Change
γ	Word Separator
ǂ	Record Mark
ǂǂ	Group Mark
ǂǂǂ	Segment Mark
✓	Tape Mark
& # @ % ¤	version commerciale
+ = ’ ( )	version FORTRAN

TABLE 5: Le code BCDIC

trous dans les rangées de chiffres [5].

Il y a eu plusieurs variantes de ce code. Deux sont données ici : la variante commerciale, utilisée pour ce genre d'application, et la variante FORTRAN, utilisée pour programmer dans ce langage. Les caractères qui diffèrent suivant les variantes sont dits duaux.

Si tous les caractères ont une représentation graphique, certains caractères étaient aussi utilisés comme caractère de commande. En particulier, certains étaient interprétés dans tous les contextes comme des commandes par les lecteur de bandes magnétiques. Ils n'aurait donc pas été stockables sur bande sans un surcodage, ce qui n'était en pratique pas mis en œuvre.

Pour plus de renseignements sur les codes pour cartes, voir [**?**, **?**, **?**, **?**, **?**]. [**?**] montre quelques propositions de code pour cartes perforées pour les lettres datant des années 30, avant que le codage des 43 caractères ne devienne quasiment universel.

3.4 EBCDIC

Au début des années 60, les insuffisances des jeux de caractères 6 bits tels que le BCDIC étaient bien connues : d'une part, ils n'avaient pas assez de caractères graphiques, donnant donc naissance à des variantes pour des usages particuliers, d'autre part ils n'avaient pas assez de caractères de contrôle, donnant donc naissance à des codes particuliers pour communiquer avec les équipements, et à la nécessité de conversions.

Une fois choisi de baser le System/360 autour d'une unité d'adressage de 8 bits, c'était l'occasion de concevoir un jeu de caractères codé qui s'affranchissait de ces limitations en profitant des 256 points de code disponibles. C'était a peu près au même moment que le développement de l'ASCII commençait, mais avec des contraintes différentes.

Pour les clients d'IBM — et donc pour IBM — la compatibilité avec BCDIC était d'une importance majeure. Ils avaient déjà beaucoup d'informations codées sur des cartes perforées. Avoir une conversion simple était une nécessité. Et respecter la structure globale aussi : une technique habituelle à l'époque était d'utiliser indépendamment les rangées de zones et de chiffres. C'est-à-dire que les caractères — ou plutôt les combinaisons de trous correspondant à ces caractères —? et A à I étaient utilisés pour les chiffres 0 à 9 combinés avec un drapeau, de même que les caractères! et J à R. Conserver le même code pour les cartes perforées pour les lettres étaient une des contraintes que les concepteurs d'EBCDIC s'étaient imposées, et conserver une équivalence simple entre les chiffres combinés avec d'autres informations en était une autre. On a l'explication de pourquoi les lettres sont ainsi codées en EBCDIC — mais par rapport au BCDIC, elles sont au moins dans l'ordre alphabétique. [6]

La prise en compte de ces considérations et d'autres considérations a fini, après quelques compromis parce que les contraintes étaient incompatibles, par donner le code

5. Il y a quelques exceptions causées par le fait que 0 est à la fois une rangée de zone et de chiffres.

6. Ces explications sont essentiellement celles de MacKenzie dans [**?**]. Robert Bemer (surnommé *le père de l'ASCII*, voir [**?**]) explique qu'en prime, le System/360 avait été conçu pour traiter l'EBCDIC et l'ASCII. Mais que des retards dans la standardisation et dans la conception des périphériques ont mené à la décision de faire d'EBCDIC le code principal. Ce choix fut par la suite tellement implicite dans la programmation que traiter de l'ASCII était en pratique impossible.

HEX			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	BIN		00				01				10				11			
			00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
		OCT	*0*	*2*	*4*	*6*	*10*	*12*	*14*	*16*	*20*	*22*	*24*	*26*	*30*	*32*	*34*	*36*
0	0000	*0*	0 (NUL)	16 (DLE)	32 (PAD)	48 (DCS)	64 (SP)	80 &	96 -	112	128	144	160	176	192 {	208 }	224 \	240 0
1	0001	*1*	1 (SOH)	17 (DC1)	33 (HOP)	49 (PUL)	65	81	97 /	113	129 a	145 j	161 ~	177	193 A	209 J	225	241 1
2	0010	*2*	2 (STX)	18 (DC2)	34 (BHP)	50 (SYN)	66	82	98	114	130 b	146 k	162 s	178	194 B	210 K	226 S	242 2
3	0011	*3*	3 (ETX)	19 (DC3)	35 (NBH)	51 (STS)	67	83	99	115	131 c	147 l	163 t	179	195 C	211 L	227 T	243 3
4	0100	*4*	4 (ST)	20 (OC)	36 (IND)	52 (CCH)	68	84	100	116	132 d	148 m	164 u	180	196 D	212 M	228 U	244 4
5	0101	*5*	5 (HT)	21 (NEL)	37 (LF)	53 (MW)	69	85	101	117	133 e	149 n	165 v	181	197 E	213 N	229 V	245 5
6	0110	*6*	6 (SSA)	22 (BS)	38 (ETB)	54 (SPA)	70	86	102	118	134 f	150 o	166 w	182	198 F	214 O	230 W	246 6
7	0111	*7*	7 (DEL)	23 (ESA)	39 (ESC)	55 (EOT)	71	87	103	119	135 g	151 p	167 x	183	199 G	215 P	231 X	247 7
8	1000	*0*	8 (EPA)	24 (CAN)	40 (HTS)	56 (SOS)	72	88	104	120	136 h	152 q	168 y	184	200 H	216 Q	232 Y	248 8
9	1001	*1*	9 (RI)	25 (EM)	41 (HTJ)	57 (SGC)	73	89	105	121 `	137 i	153 r	169 z	185	201 I	217 R	233 Z	249 9
A	1010	*2*	10 (SS2)	26 (PU2)	42 (VTS)	58 (SCI)	74 ¢	90 !	106 ¦	122 :	138	154	170	186	202	218	234	250
B	1011	*3*	11 (VT)	27 (SS3)	43 (PLD)	59 (CSI)	75 .	91 $	107 ,	123 #	139	155	171	187	203	219	235	251
C	1100	*4*	12 (FF)	28 (IS4)	44 (PLU)	60 (DC4)	76 <	92 *	108 %	124 @	140	156	172	188	204	220	236	252
D	1101	*5*	13 (CR)	29 (IS3)	45 (ENQ)	61 (NAK)	77 (	93)	109 _	125 ’	141	157	173	189	205	221	237	253
E	1110	*6*	14 (SO)	30 (IS2)	46 (ACQ)	62 (PM)	78 +	94 ;	110 >	126 =	142	158	174	190	206	222	238	254
F	1111	*7*	15 (SI)	31 (IS1)	47 (BEL)	63 (SUB)	79 |	95 ¬	111 ?	127 "	143	159	175	191	207	223	239	255 (APC)
		OCT	*1*	*3*	*5*	*7*	*11*	*13*	*15*	*17*	*21*	*23*	*25*	*27*	*31*	*33*	*35*	*37*

TABLE 6: Le code EBCDIC

montré en table **??**.

Malgré l'objectif de supprimer les variantes, celles d'EBCDIC sont nombreuses. Mais un progrès par rapport à BCDIC, un site donné pouvait généralement trouver une variante adéquate à l'ensemble de ses applications : les variantes étant plutôt dues à la localisation. Dans la variété des codes EBCDIC, il est à remarquer que certains caractères courants n'ont pas toujours le même code quand ils sont présents.

L'histoire des codes BCDIC et EBCDIC jusque 1968 est décrite dans [**?**] avec beaucoup de détails sur les contraintes et les raisonnements ayant guidés les choix.

	HEX		0	1	2	3	4	5	6	7
	BIN		000	001	010	011	100	101	110	111
		OCT	*0*	*2*	*4*	*6*	*10*	*12*	*14*	*16*
0	0000	*0*	0 (NUL)	16 (DLE)	32 (SP)	48 0	64 @	80 P	96 `	112 p
1	0001	*1*	1 (SOH)	17 (DC1)	33 !	49 1	65 A	81 Q	97 a	113 q
2	0010	*2*	2 (STX)	18 (DC2)	34 "	50 2	66 B	82 R	98 b	114 r
3	0011	*3*	3 (LTX)	19 (DC3)	35 #	51 3	67 C	83 S	99 c	115 s
4	0100	*4*	4 (EOT)	20 (DC4)	36 $	52 4	68 D	84 T	100 d	116 t
5	0101	*5*	5 (ENO)	21 (NAK)	37 %	53 5	69 E	85 U	101 e	117 u
6	0110	*6*	6 (ACK)	22 (SYN)	38 &	54 6	70 F	86 V	102 f	118 v
7	0111	*7*	7 (BEL)	23 (ETB)	39 ’	55 7	71 G	87 W	103 g	119 w
8	1000	*0*	8 (BS)	24 (CAN)	40 (	56 8	72 H	88 X	104 h	120 x
9	1001	*1*	9 (HT)	25 (EM)	41)	57 9	73 I	89 Y	105 i	121 y
A	1010	*2*	10 (LF)	26 (SUB)	42 *	58 :	74 J	90 Z	106 j	122 z
B	1011	*3*	11 (VT)	27 (ESC)	43 +	59 ;	75 K	91 [	107 k	123 {
C	1100	*4*	12 (FF)	28 (FS)	44 ,	60 <	76 L	92 \	108 l	124 |
D	1101	*5*	13 (CR)	29 (GS)	45 -	61 =	77 M	93]	109 m	125 }
E	1110	*6*	14 (SO)	30 (RS)	46 .	62 >	78 N	94 ^	110 n	126 ~
F	1111	*7*	15 (SI)	31 (US)	47 /	63 ?	79 O	95 _	111 o	127 (DEL)
		OCT	*1*	*3*	*5*	*7*	*11*	*13*	*15*	*17*

TABLE 7: Le code ASCII

3.5 ASCII

Ce code (voir table **??**) est un code 7 bit développé aux États-Unis durant les années 60 (avec des étapes en 63, 65, 67 et 68 ; la version présentée ici est celle de 68 d’après [**?**] [7]. Ses concepteurs étaient conscients qu’il était peu probable que des systèmes utilisent exactement 7 bits de manière interne [**?**, p. 217]. Le choix d’un code sur 7 bits a cependant été fait car il satisfaisait les besoins de l’utilisation principale prévue : l’échange de donnée. Un code à 8 bits aurait été perçu comme imposant un coût inutile à la majorité et en prime que l’ajout d’un huitième bit ne permettait plus l’utilisation aisée de certaines techniques comme les rubans perforés (ceux-ci permettaient jusqu’à 8 bits mais

7. Mais cette référence utilise ¦ plutôt que |. Voir **??**.

un de ceux-ci était utilisé comme bit de parité).

Par la suite, il fut normalisé par l'ISO (sous le nom ISO 646, ECMA a publié une norme techniquement équivalente sous le nom ECMA 6 qui a l'avantage d'être disponible publiquement [**?**]). Cette norme prévoit la possibilité pour des variantes nationales de changer certains caractères pour correspondre à des nécessités nationales (par exemple l'ajout de lettres accentuées). De même, les variantes nationales peuvent prévoir que l'utilisation de séquences de bytes pour coder certains caractères (par exemple la variante française prévoyait l'utilisation de la séquence « ~(BS) a » pour le caractère « â »).

Les caractères pouvant avoir une définition nationale sont #, $, @, [, \,], ^, `, {, |, }, ~. De plus les choix pour # et $ sont restreints (# peut être remplacé par £, et $ par ¤ ; ce dernier caractère — appelé « symbole monétaire » — a d'ailleurs été présent dans certaines versions de référence).

Il y eut plusieurs révisions de cette norme. La plus importante fut vraisemblablement pour la mettre en conformité avec la structure de ISO 2022 (voir **??**) en enlevant la description des caractères de contrôle pour la confier à d'autres normes (ISO 6429 alias ECMA 48 [**?**] par exemple). Dans cette forme, ce jeu de caractères ne comporte plus que 94 caractères, ceux de codets 33 à 126 (l'espace et DEL font partie de ISO 2022).

L'existence de variantes nationales — parfois plusieurs variantes par pays — a posé suffisemment de problèmes pratiques pour que la France ait fini par retirer ses variantes et adopter la version de référence comme variante officielle.

Ce jeu de caractères a servi de base à beaucoup d'autres qui ont rempli les 128 positions laissées libres si on le code avec une unité faisant 8 bits ; ces jeux sont souvent appelés « ASCII étendu ». Il faut être bien conscient en utilisant cette appellation qu'elle ne désigne pas un charset bien fixé.

L'histoire de l'ASCII est décrite par [**?**] et [**?**] avec beaucoup de détails, entre autres sur les différentes versions intermédiaires et les raisons des changements d'une à l'autre. Même si [**?**] passe très vite de la norme de 1963 à celle de 1967, il décrit l'établissement d'un code pour carte perforée — basé sur le code Hollerith.

3.6 ISO 2022

La norme ISO 2022 (disponible plus faciment en tant que norme ECMA 35 [**?**]) décrit un mécanisme de codage permettant de coder plusieurs jeux de caractères codés respectant une certaine structure dans un même flux. Il faut noter que cette norme ne décrit pas en elle-même un code, mais une structure pour des codes, avec des choix qui doivent être fait pour avoir un code spécifié complètement.

Deux types de bytes sont possibles : 7 bits et 8 bits.

Trois types de charsets sont codables :

- des jeux de 32 caractères de commandes ayant les codets 0 à 31.
- des jeux d'au maximum 94^n caractères graphiques codés par des codets ayant pour valeur $(x_1 \dots x_n)$ où les x_i sont compris entre 33 et 126.
- des jeux d'au maximum 96^n caractères graphiques codés par des codets ayant pour valeur $(x_1 \dots x_n)$ où les x_i sont compris entre 32 et 127.

Un code a au maximum 2 charsets composés de caractères de commande privilégiés appelé C0 et C1, et 4 charsets composés de caractères graphiques privilégiés appelés G0,

G1, G2 et G3.

Dans un code utilisant des bytes de 7 bits, deux charsets sont accessibles simultanément : un charset composés de caractère de commandes et appelé CL et un charset composés de 94^n caractères graphiques appelé GL.

– un caractère de CL est codé par un byte ayant pour valeur son codet ;
– CL doit avoir un caractère de commande (ESCAPE) de codet 27 ;
– un caractère de GL est codé par n bytes ayant pour valeur x_i ;
– le caractère ESPACE est codé par le codet 32 ;
– le caractère de contrôle (DEL) est codé par le codet 127.

Dans un code utilisant des bytes de 8 bits, quatres charsets sont accessibles simultanément : deux charsets composés de caractères de contrôles appelés CL et CR, un charsets de 94^n caractères graphiques appelé GL, un charset de 94^n ou de 96^n caractères graphiques appelé GR.

– un caractère de CL est codé par un byte ayant pour valeur son codet ;
– CL doit avoir un caractère de commande (ESC) de codet 27 ;
– un caractère de CR est codé par un byte ayant pour valeur 128 plus son codet ;
– CR ne peut pas avoir le caractère de commande (ESC) ;
– un caractère de GL est codé par n bytes ayant pour valeur x_i ;
– le caractère ESPACE est codé par le codet 32 ;
– le caractère de contrôle (DEL) est codé par le codet 127 ;
– un caractère de GR est codé par n bytes ayant pour valeur $128 + x_i$.

Il y a des fonctions de commandes pour
– indiquer que CL désigne C0 ou C1,
– indiquer que CR désigne C0 ou C1,
– indiquer que GL désigne G0, G1, G2 ou G3,
– indiquer que GR désigne G0, G1, G2 ou G3,
– modifier les charsets que C0, C1, G0, G1, G2 et G3 désignent.

Ces fonctions sont accessibles soit par des séquences de bytes commançant par (ESC) ; elles peuvent aussi être assignées à un caractère dans des charsets de commandes.

Un mécanisme d'enregistrement permet de s'assurer que la séquence de commande pour choisir un charset donné est la même dans tous les codes respectant ISO 2022 (la liste des charsets enregistrés et les séquences pour les choisir est disponible en ligne, voir [**?**]).

3.7 ISO 8859

La série de normes ISO 8859 (dont la publication s'étable de 1987 pour ISO 8859-1 à 2001 pour ISO 8859-16) normalise des codes respectant la structure de ISO 2022 ayant des caractéristiques communes :

– le byte fait 8 bits ;
– CL référence toujours C0, CR référence toujours C1 ; les charsets précis sont non spécifiés ;
– GL référence toujours G0 qui correspond à l'ASCII ;
– GR référence toujours G1 qui varie suivant les codes ;
– il n'y a pas de G2 ou de G3, on n'utilise pas de fonctions pour modifier C0, C1, G0,

HEX			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	BIN		00				01				10				11			
			00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
		OCT	*0*	*2*	*4*	*6*	*10*	*12*	*14*	*16*	*20*	*22*	*24*	*26*	*30*	*32*	*34*	*36*
0	0000	*0*	0	16	32 (SP)	48 0	64 @	80 P	96 `	112 p	128	144	160 (NBS)	176 °	192 À	208 Ð	224 à	240 ð
1	0001	*1*	1	17	33 !	49 1	65 A	81 Q	97 a	113 q	129	145	161 ¡	177 ±	193 Á	209 Ñ	225 á	241 ñ
2	0010	*2*	2	18	34 "	50 2	66 B	82 R	98 b	114 r	130	146	162 ¢	178 ²	194 Â	210 Ò	226 â	242 ò
3	0011	*3*	3	19	35 #	51 3	67 C	83 S	99 c	115 s	131	147	163 £	179 ³	195 Ã	211 Ó	227 ã	243 ó
4	0100	*4*	4	20	36 $	52 4	68 D	84 T	100 d	116 t	132	148	164 ¤	180 ´	196 Ä	212 Ô	228 ä	244 ô
5	0101	*5*	5	21	37 %	53 5	69 E	85 U	101 e	117 u	133	149	165 ¥	181 µ	197 Å	213 Õ	229 å	245 õ
6	0110	*6*	6	22	38 &	54 6	70 F	86 V	102 f	118 v	134	150	166 ¦	182 ¶	198 Æ	214 Ö	230 æ	246 ö
7	0111	*7*	7	23	39 ’	55 7	71 G	87 W	103 g	119 w	135	151	167 §	183 ·	199 Ç	215 ×	231 ç	247 ÷
8	1000	*0*	8	24	40 (	56 8	72 H	88 X	104 h	120 x	136	152	168 ¨	184 ¸	200 È	216 Ø	232 è	248 ø
9	1001	*1*	9	25	41)	57 9	73 I	89 Y	105 i	121 y	137	153	169 ©	185 ¹	201 É	217 Ù	233 é	249 ù
A	1010	*2*	10	26	42 *	58 :	74 J	90 Z	106 j	122 z	138	154	170 ª	186 º	202 Ê	218 Ú	234 ê	250 ú
B	1011	*3*	11	27	43 +	59 ;	75 K	91 [	107 k	123 {	139	155	171 «	187 »	203 Ë	219 Û	235 ë	251 û
C	1100	*4*	12	28	44 ,	60 <	76 L	92 \	108 l	124 |	140	156	172 ¬	188 ¼	204 Ì	220 Ü	236 ì	252 ü
D	1101	*5*	13	29	45 -	61 =	77 M	93]	109 m	125 }	141	157	173 (SHY)	189 ½	205 Í	221 Ý	237 í	253 ý
E	1110	*6*	14	30	46 .	62 >	78 N	94 ^	110 n	126 ~	142	158	174 ®	190 ¾	206 Î	222 Þ	238 î	254 þ
F	1111	*7*	15	31	47 /	63 ?	79 O	95 _	111 o	127 (DEL)	143	159	175 ¯	191 ¿	207 Ï	223 ß	239 ï	255 ÿ
		OCT	*1*	*3*	*5*	*7*	*11*	*13*	*15*	*17*	*21*	*23*	*25*	*27*	*31*	*33*	*35*	*37*

TABLE 8: Le code ISO 8859-1 ou Latin 1

G1, G2, G3, CL, CR, GL ou G4.

Seul varie donc le contenu de G1. Le contenu de G1 est choisi pour permettre de coder des textes dans des ensembles de langues différents. Deux variantes sont pertinentes pour le français :

- ISO 8859-1 (appelé aussi latin-1, voir **??**) qui était sensé être le charset à utilisé pour le français mais qui n'a pas 4 caractères nécessaires : œ, Œ et Ÿ. Il était cependant le meilleur choix jusqu'à l'introduction du second.
- ISO 8859-15 (appelé aussi latin-9 et même latin-0, voir **??**) qui corrige ce défaut et quelques autres de ISO 8859-1 et introduit le caractère €. C'est le charset sur 8 bits à préferer pour le français.

HEX			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	BIN		00				01				10				11			
			00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
		OCT	*0*	*2*	*4*	*6*	*10*	*12*	*14*	*16*	*20*	*22*	*24*	*26*	*30*	*32*	*34*	*36*
0	0000	*0*	0	16	32 (SP)	48 0	64 @	80 P	96 `	112 p	128	144	160 (NBS)	176 °	192 À	208 Đ	224 à	240 ð
1	0001	*1*	1	17	33 !	49 1	65 A	81 Q	97 a	113 q	129	145	161 ¡	177 ±	193 Á	209 Ñ	225 á	241 ñ
2	0010	*2*	2	18	34 "	50 2	66 B	82 R	98 b	114 r	130	146	162 ¢	178 ²	194 Â	210 Ò	226 â	242 ò
3	0011	*3*	3	19	35 #	51 3	67 C	83 S	99 c	115 s	131	147	163 £	179 ³	195 Ã	211 Ó	227 ã	243 ó
4	0100	*4*	4	20	36 $	52 4	68 D	84 T	100 d	116 t	132	148	164 €	180 Ž	196 Ä	212 Ô	228 ä	244 ô
5	0101	*5*	5	21	37 %	53 5	69 E	85 U	101 e	117 u	133	149	165 ¥	181 µ	197 Å	213 Õ	229 å	245 õ
6	0110	*6*	6	22	38 &	54 6	70 F	86 V	102 f	118 v	134	150	166 Š	182 ¶	198 Æ	214 Ö	230 æ	246 ö
7	0111	*7*	7	23	39 ’	55 7	71 G	87 W	103 g	119 w	135	151	167 §	183 ·	199 Ç	215 ×	231 ç	247 ÷
8	1000	*0*	8	24	40 (	56 8	72 H	88 X	104 h	120 x	136	152	168 š	184 ž	200 È	216 Ø	232 è	248 ø
9	1001	*1*	9	25	41)	57 9	73 I	89 Y	105 i	121 y	137	153	169 ©	185 ¹	201 É	217 Ù	233 é	249 ù
A	1010	*2*	10	26	42 *	58 :	74 J	90 Z	106 j	122 z	138	154	170 ª	186 º	202 Ê	218 Ú	234 ê	250 ú
B	1011	*3*	11	27	43 +	59 ;	75 K	91 [	107 k	123 {	139	155	171 «	187 »	203 Ë	219 Û	235 ë	251 û
C	1100	*4*	12	28	44 ,	60 <	76 L	92 \	108 l	124 |	140	156	172 ¬	188 Œ	204 Ì	220 Ü	236 ì	252 ü
D	1101	*5*	13	29	45 -	61 =	77 M	93]	109 m	125 }	141	157	173 (SHY)	189 œ	205 Í	221 Ý	237 í	253 ý
E	1110	*6*	14	30	46 .	62 >	78 N	94 ^	110 n	126 ~	142	158	174 ®	190 Ÿ	206 Î	222 Þ	238 î	254 þ
F	1111	*7*	15	31	47 /	63 ?	79 O	95 _	111 o	127 (DEL)	143	159	175 ¯	191 ¿	207 Ï	223 ß	239 ï	255 ÿ
		OCT	*1*	*3*	*5*	*7*	*11*	*13*	*15*	*17*	*21*	*23*	*25*	*27*	*31*	*33*	*35*	*37*

TABLE 9: Le code ISO 8859-15 ou Latin 9

Pour référence, voici les langues destinées à être codées par les différentes variantes :

ISO 8859-1 (latin-1) langues d’Europe occidentale ;

ISO 8859-2 (latin-2) langues d’Europe de l’Est ;

ISO 8859-3 (latin-3) langues d’Europe du Sud ;

ISO 8859-4 (latin-4) langues d’Europe du Nord ;

ISO 8859-5 alphabet cyrillique ;

ISO 8859-6 alphabet arabe ;

ISO 8859-7 alphabet grec ;

ISO 8859-8 alphabet hébraïque ;

ISO 8859-9 (latin-5)langues d'Europe occidentale, avec les caractères nécessaires au turc à la place de ceux nécessaires à l'islandais ;

ISO 8859-10 (latin-6) langues d'Europe du Nord, y compris l'islandais ;

ISO 8859-11 alphabet thaï ;

ISO 8859-12 jamais défini ;

ISO 8859-13 (latin-7) langues baltes et polonais ;

ISO 8859-14 (latin-8) langues celtiques ;

ISO 8859-15 (latin-9) langues d'Europe occidentale, avec € et les caractères pour le français manquant dans ISO 8859-1 ;

ISO 8859-16 (latin-10) langue d'Europe centrale, avec € et les caractères pour le français (même ceux manquant dans ISO 8859-1) et le roumain (qui n'était traité par aucun autre code de ISO 8859).

3.8 Microsoft

Les normes ISO 8859 furent loin d'être les premiers codes étendant l'ASCII. Certains l'on fait bien avant en assignant des caractères graphiques aux caractères de commande. La plupart ont assigné des caractères graphiques en étendant les codets sur 8 bits. Parmis ceux-ci citons en deux :

- CP 437 était le charset (CP pour *code page*, le terme pour charset du jargon d'IBM repris par Microsoft) hardcodé dans l'IBM PC ; c'est un pot pourri de caractères — des lettres accentuées, des lettres grecques, des caractères permettant de faire des filets, etc. — dont le choix relève d'une logique mystérieuse (il y à un é, un è, un É mais pas de È par exemple) ;
- CP 850 a été introduit quand Microsoft a introduit la possibilité de changer la fonte en DOS ; ce charset est une variante de CP 437 comportant des caractères accentués en remplacement d'une partie des caractères (principalement certains filets, les lettres grecques et certains symboles mathématiques), ce qui permet d'écrire les langues d'Europe occidentale.

Lors de l'introduction de Windows, Microsoft a décidé de revoir sa gestion des charsets. Les caractères permettant de faire des filets n'étaient pas plus utiles avec Windows cela permettaient d'introduire les caractères nécessaires aux différentes langues. Pour la version destinée à l'Europe occidentale, Microsoft est partit d'un brouillon de ISO 8859-1 qui était à l'époque en cours de normalisation et a ajouté des caractères graphiques dans la zone CR. Le charset résultant (voir **??**) est appelé CP 1252 ou *Windows Latin 1*. Il a continué à évoluer, d'une part en suivant les modifications faites à ISO 8859-1, d'autre part en ajoutant de nouveaux caractères dans la zone CR.

Il y a naturellement d'autres pages de code utilisées par Windows, parmis lesquelles :

CP 1250 *Windows Latin 2*, basée sur ISO 8859-2 mais avec des modifications en plus de l'ajout de caractères graphiques dans la zone CR ;

CP 1251 *Windows cyrillique* ;

CP 1253 *Windows grec* ;

HEX			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	BIN		00				01				10				11			
			00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
		OCT	*0*	*2*	*4*	*6*	*10*	*12*	*14*	*16*	*20*	*22*	*24*	*26*	*30*	*32*	*34*	*36*
0	0000	*0*	0	16	32 (SP)	48 0	64 @	80 P	96 `	112 p	128 €	144	160 (NBS)	176 °	192 À	208 Ð	224 à	240 ð
1	0001	*1*	1	17	33 !	49 1	65 A	81 Q	97 a	113 q	129	145 ‘	161 ¡	177 ±	193 Á	209 Ñ	225 á	241 ñ
2	0010	*2*	2	18	34 "	50 2	66 B	82 R	98 b	114 r	130 ‚	146 ’	162 ¢	178 ²	194 Â	210 Ò	226 â	242 ò
3	0011	*3*	3	19	35 #	51 3	67 C	83 S	99 c	115 s	131 *ƒ*	147 “	163 £	179 ³	195 Ã	211 Ó	227 ã	243 ó
4	0100	*4*	4	20	36 $	52 4	68 D	84 T	100 d	116 t	132 „	148 ”	164 ¤	180 ´	196 Ä	212 Ô	228 ä	244 ô
5	0101	*5*	5	21	37 %	53 5	69 E	85 U	101 e	117 u	133 …	149 •	165 ¥	181 µ	197 Å	213 Õ	229 å	245 õ
6	0110	*6*	6	22	38 &	54 6	70 F	86 V	102 f	118 v	134 †	150 –	166 ¦	182 ¶	198 Æ	214 Ö	230 æ	246 ö
7	0111	*7*	7	23	39 ’	55 7	71 G	87 W	103 g	119 w	135 ‡	151 —	167 §	183 ·	199 Ç	215 ×	231 ç	247 ÷
8	1000	*0*	8	24	40 (	56 8	72 H	88 X	104 h	120 x	136 ^	152 ~	168 ¨	184 ¸	200 È	216 Ø	232 è	248 ø
9	1001	*1*	9	25	41)	57 9	73 I	89 Y	105 i	121 y	137 ‰	153 ™	169 ©	185 ¹	201 É	217 Ù	233 é	249 ù
A	1010	*2*	10	26	42 *	58 :	74 J	90 Z	106 j	122 z	138 Š	154 š	170 ª	186 º	202 Ê	218 Ú	234 ê	250 ú
B	1011	*3*	11	27	43 +	59 ;	75 K	91 [	107 k	123 {	139 ‹	155 ›	171 «	187 »	203 Ë	219 Û	235 ë	251 û
C	1100	*4*	12	28	44 ,	60 <	76 L	92 \	108 l	124 |	140 Œ	156 œ	172 ¬	188 ¼	204 Ì	220 Ü	236 ì	252 ü
D	1101	*5*	13	29	45 -	61 =	77 M	93]	109 m	125 }	141	157	173 (SHY)	189 ½	205 Í	221 Ý	237 í	253 ý
E	1110	*6*	14	30	46 .	62 >	78 N	94 ^	110 n	126 ~	142 Ž	158	174 ®	190 ¾	206 Î	222 Þ	238 î	254 þ
F	1111	*7*	15	31	47 /	63 ?	79 O	95 _	111 o	127 (DEL)	143 ž	159	175 ¯	191 ¿	207 Ï	223 ß	239 ï	255 ÿ
		OCT	*1*	*3*	*5*	*7*	*11*	*13*	*15*	*17*	*21*	*23*	*25*	*27*	*31*	*33*	*35*	*37*

TABLE 10: Le code CP 1252 ou MS ANSI

CP 1255 *Windows hébraïque*;

CP 1256 *Windows arabe*;

CP 1257 *Windows pays baltes.*

Deux termes sont aussi utilisés pour désigner des charsets sous Windows :

OEM ce terme désigne soit CP 457, soit n'importe quelle page de code utilisée sous DOS ;

ANSI ce terme désigne soit CP 1252, soit n'importe quelle page de code 8 bits utilisées par Windows.

3.9 ISO 10646 et Unicode

Vers la fin des années 80 la situation en ce qui concerne le codage des caractères était qu'on avait des jeux de caractères capables de traiter des langues ou des groupes de langues, mais aucun capable de représenter tous les caractères possibles. ISO 2022 fournissait bien un structure capable de représenter des caractères provenant de plusieurs charset, mais cette structure avait été conçue dans une optique de communication et était mal adaptée aux contextes où il faut manipuler des caractères de différentes sources. Outre des problèmes pratiques de mise en oœvre, il faut savoir quels caractères de jeux différents se correspondent.

Deux projets commencèrent donc à étudier la possibilé d'un jeu de caractère universel : Unicode et ISO 10646.

Unicode, produit d'un consortium de compagnies privées, est parti sur un jeu codé sur 16 bits et avec un objectif plus large. En plus d'un codage, Unicode définit des choses tel que les algorithmes à utiliser pour découper un texte en ligne quand on mélange des écritures s'écrivant dans des directions différentes.

La première proposition pour ISO 10646 était compatible avec la structure des jeux de ISO 2022 (mais ne la respectait pas complètement) : les codets n'utilisent aucun byte dans les zones de contrôles (0x00 à 0x2F et 0x80 à 0x9F). Cette proposition fut rejetée car les industriels étaient opposés à la gestion de deux jeux universels, avec les coûts et les problèmes qui s'en seraient suivit.

Il s'en est suivit une unification partielle avec Unicode. ISO 10646 s'occupant de l'assignation des caractères. En principe, ISO 10646 est aussi plus ouvert sur la possibilité d'utilisation de codets non représentable en UTF-16 et le codage de langues anciennes ; je n'ai pas connaissance d'effets pratiques de ces différences.

Unicode garde en propre certains aspects plus stricts dans son objectif, la définition d'équivalence entre des formes précomposées et leur décomposition — nous verrons que le codage prévoit un caractère « é », un caractère « e » et un caractère accent aigu qui modifie le caractère précédent ; pour Unicode « é » et « e » avec un accent aigu sont équivalent, cette équivalence est en dehors de ce qui est traité par ISO 10646 — et les autres points d'Unicode non relatifs au codage.

Par la suite, nous parlerons d'Unicode sans préciser si le point est propre à Unicode ou bien partagé avec ISO 10646.

Structure

Mécanismes de codage

Unicode définit une série de mécanismes de codage pour représenter les différents codets. Ces mécanismes de codage sont tous non-modaux ; ils diffèrent par la taille du byte utilisé et le sous-ensemble des codets qu'il est possible d'utiliser (certain ne permettent de représenter que les codets du plan de base).

UCS-32 Le byte utilisé a 32 bits et les codets sont simplement représentés par un byte de leur valeur. S'il faut en plus sérialiser en octets, on peut indiquer par un suffixe « BE » ou « LE » si l'octet le plus significatif est en premier ou en dernier.

UCS-16 Le byte utilisé a 16 bits et les codets sont simplement représentés par un byte de leur valeur et on ne représente que les codets du plan de base. S'il faut en plus sérialiser en octets, on peut indiquer par un suffixe « BE » ou « LE » si l'octet le plus significatif est en premier ou en dernier.

UTF-16 Le byte utilisé a 16 bits et les codets sont simplement représentés par un byte de leur valeur. Les codets en dehors du plan de base sont représentés par une paire de seizets (dits d'indirection). En effet, une zone de 2048 codets est volontairement laissée non assignée dans le plan de base à cet effet. Unicode est limité aux codets représentables en UTF-16 (de 0 à $2^{20} + 2^{16} - 1$), ISO 10646 non. S'il faut en plus sérialiser en octets, on peut indiquer par un suffixe « BE » ou « LE » si l'octet le plus significatif est en premier ou en dernier.

UTF-8 Le byte utilisé a 8 bits. Les codets sont représentés par 1 à 6 octets (Unicode limite ce nombre à 4 et impose qu'en avoir plus soit une erreur). Le codage utilisé permet d'avoir des représentations multiples pour certains caractères, il faut utiliser la plus courte possible.

4 Au sujet de quelques caractères

4.1 La barre oblique inversée

4.2 Les barres verticales

Le codet 124 de l'ASCII (et des jeux dérivés dont le jeux 8 bits d'ISO 8859 et Unicode) est une barre verticale « | ». Il existe aussi un caractère représentant une barre vertical brisée « ¦ » et les deux caractères sont disponibles simultanément dans EBCDIC, ISO-8859-1, Unicode et peut-être d'autres jeux encore.

Malgré cela, les fontes représentent souvent la barre verticale brisée, et ce parfois même quand elles ont aussi un glyphe pour le caractère barre verticale brisée (le même glyphe étant utilisé alors pour les deux caractères, ce qui peut causer de la confusion).

La raison est que SHARE, le groupe des utilisateurs d'IBM, insista fortement — allant jusqu'à menacer de boycotter la norme si on ne leur donnait pas satisfaction — pour avoir les caractères « | » et « ¬ » dans la zone centrale (qui sert de base au sous-ensemble codé sur 6 bits) et en dehors des caractères à usage nationaux. Plutôt que de remettre en cause les compromis qui venaiet d'être acquis au niveau international, X3.2 — le comité américain en charge des jeux de caractères — changea la représentation de la ligne verticale en ¦ et ajouta des notes indiquant qu'il était possible de styliser le point d'exclamation « ! » en « | » et l'accent circonflexe « ^ » en « ¬ » [**?**]. Ces lattitudes furent supprimées lors de l'édition de 1977 de l'ASCII, et même avant des tables officielle de l'ASCII on montré la barre verticale pleine (par exemple dans l'enregistrement de l'ASCII dans le répertoire internationnal des jeux de caractères codé [**?**]).

Un autre effet de leur action est dans le code pour carte perforée. Les schémas de trous utilisés pour coder les caractères graphiques l'ASCII sur des cartes perforées sont les mêmes que pour les caractères équivalent d'EBCDIC, à quelques exceptions près Quadi :punch,charsethist :

– 12-8-2 est utilisé pour « ç » en EBCDIC, pour « [» en ASCII ;
– 11-8-2 est utilisé pour « ! » en EBCDIC, pour «] » en ASCII ;
– 12-8-7 est utilisé pour « | » en EBCDIC, pour « ! » en ASCII ;
– 11-8-7 est utilisé pour « ¬ » en EBCDIC, pour « ^ » en ASCII ;

4.3 Les lettres manquantes d'ISO 8859-1

[**?**]

Glossaire

byte (*byte*) ([**?**] utilise *multiplet* qui est rarement employé dans l'usage courant)

1/ chaîne composée d'un certain nombre de bits traitée comme un tout et représentant habituellement un caractère ou une partie de caractère [**?**].

2/ la plus petite unité adressable par un ordinateur. En particulier quand le mot-machine n'est pas la plus petite unité adressable.

3/ chaîne de bits formée de 8 bits (voir *octet*).

caractère (*character*) Élément d'un ensemble employé pour constituer, représenter ou gérer des données [**?**].

code (*code*, *coding scheme*) ensemble de règles établissant une correspondance entre les éléments d'un premier ensemble et ceux d'un second ensemble [**?**].

codage 1/ transformation d'une représentation en une autre.

2/ règles utilisée pour effectuer cette transformation (voir *code*).

codet (*code value*, *code element*) résultat de l'application d'un code à un élément d'un jeu codé [**?**].

décodage opération inverse de l'encodage.

encodage 1/ transformation d'une représentation, considérée comme principale, en une autre.

2/ règles utilisées pour effectuer cette transformation (voir *code*).

jeu de caractères (*character set*) ensemble fini de caractères considéré comme complet pour un usage particulier.

jeu de caractères codé (*coded character set*) ensemble de caractères mis en correspondance avec un autre ensemble d'éléments selon un code [**?**].

multiplet (*byte*) voir *byte*.

octet chaîne de bits formée de 8 bits.

seizet chaîne de bits formée de 16 bits.

Lexique français-anglais

boutisme endianness
byte byte
caractère character
code code, coding scheme
codet code value, code element, code point
fonte font
gros-boutien big-endian
gros-boutiste big-endian
jeu de caractères character set
jeu de caractères codé coded character set
multiplet byte
numéro de caractère code point
page de code code page
petit-boutien little-endian
petit-boutiste little-endian
point de code code point
police font

Lexique anglais-français

big-endian gros-boutiste, gros-boutien

byte multiplet, byte

character caractère

character set jeu de caractères

code code

coded character set jeu de caractères codé

code element codet

code page page de code

code point numéro de caractère, codet, point de code

code value codet

coding scheme code

endianness boutisme

font fonte, police

little-endian petit-boutiste, petit-boutien

Voir aussi [**?**], un lexique anglais-français du vocabulaire d'Unicode.

Bibliographie

[Anda] P. Andries. Lexique unicode. [Online]. Available : http://hapax.qc.ca/dunod/Unicode_Lexique.pdf

[Andb] ——. Unicode et iso 10646 en français. [Online]. Available : http://hapax.qc.ca

[And96] J. André, "ISO Latin-1, norme de codage des caractères européens ? trois caractères français en sont absents !" *Cahiers GUTenberg*, no. 25, 11 1996. [Online]. Available : http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/25-andre.pdf

[And02] ——, "Caractères, codage et normalisation : de Chappe à Unicode," *Document numérique*, vol. 6, no. 3, 2002.

[And08] P. Andries, *Unicode 5.0 en pratique*. Dunod, 2008.

[Bema] B. Bemer. EDCDIC and the P-Bit. [Online]. Available : http://www.bobbemer.com/P-BIT.HTM

[Bemb] ——. The great curly brace trace chase. [Online]. Available : http://www.bobbemer.com/BRACES.HTM

[Bemc] ——. How ASCII got its backslash. [Online]. Available : http://www.bobbemer.com/BACKSLASH.HTM

[Bemd] ——. A proposal for character code compatibility. [Online]. Available : http://www.bobbemer.com/ESCAPE0.HTM

[Ber09] D. Bernard, 2009, communication personnelle.

[Bra] M. Brandel. 1963 : ASCII debuts. [Online]. Available : http://www.bobbemer.com/brandela.htm

[Cha] C. Chatenet. Le télégraphe chappe. [Online]. Available : http://chappe.ec-lyon.fr

[Dep34] P. O. E. Department, *Baudot Multiplex Type-printing System*. His Majesty's Stationery Office, 1934. [Online]. Available : http://www.samhallas.co.uk/repository/telegraph/b6_baudot_multiplex.pdf

[ECMA6] *7-Bit coded Character Set*, ECMA Std. 6, 1991, technically equivalent to ISO/IEC 646. [Online]. Available : http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-006.pdf

[ECMA7] *Representation of the Standard ECMA-6 (7 bit Code) on Punched Cards*, ECMA Std. 7, 1963. [Online]. Available : http://www.ecma-international.org/publications/files/ECMA-ST-WITHDRAWN/ECMA-7,1stEdition,April1965.pdf

[ECMA35] *Character Code Structure and Extension Techniques*, ECMA Std. 35, 1994, technically equivalent to ISO/IEC 2022. [Online]. Available : http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-035.pdf

[ECMA48] *Control Functions for Coded Character Sets*, ECMA Std. 48, 1991, technically equivalent to ISO/IEC 6429. [Online]. Available : http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-048.pdf

[Fis] E. Fischer, "The evolution of character codes, 1874-1968." [Online]. Available : http://www.transbay.net/~enf/ascii/ascii.pdf

[Har04] Y. Haralambous, *Fontes & codages*. O'Reilly, 2004.

[ISO-IR] International register of coded character sets. ISO. [Online]. Available : http://www.itscj.ipsj.or.jp/ISO-IR

[ISO646] *Information technology – ISO 7 bit coded character set for information interchange*, ISO Std. ISO/IEC 646 :1991, Équivalent techniquement à la norme ECMA 6, plus facile à se procurer.

[ISO2022] *Information technology – Character code structure and extension techniques*, ISO Std. ISO/IEC 2022 :1994, Équivalent techniquement à la norme ECMA 35, plus facile à se procurer.

[ISO2382-4] *Technologies de l'information – Vocabulaire, Partie 4 : Organisation des données*, ISO/IEC Std. 2382-4 :1999(E/F). [Online]. Available : http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

[ISO2382-5] *Technologies de l'information – Vocabulaire, Partie 5 : Représentation des données*, ISO/IEC Std. 2382-5 :1999(E/F). [Online]. Available : http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

[ISO4873] *Information technology – ISO 8-bit code for information interchange – Structure and rules for implementation*, ISO Std. ISO/IEC 4873 :1991, Équivalent techniquement à la norme ECMA 43, plus facile à se procurer.

[ISO6429] *Information technology – Control functions for coded character sets*, ISO Std. ISO/IEC 6429 :1992, Équivalent techniquement à la norme ECMA 48, plus facile à se procurer.

[ISO8859-1] *Information technology – 8-bit single-byte coded graphic character sets – Part 1 : Latin alphabet No. 1*, ISO Std. ISO/IEC 8859-1 :1998, Équivalent techniquement à une partie de la norme ECMA 94, plus facile à se procurer.

[ISO8859-2] *Information technology – 8-bit single-byte coded graphic character sets – Part 2 : Latin alphabet No. 2*, ISO Std. ISO/IEC 8859-2 :1999, Équivalent techniquement à une partie de la norme ECMA 94, plus facile à se procurer.

[ISO8859-3] *Information technology – 8-bit single-byte coded graphic character sets – Part 3 : Latin alphabet No. 3*, ISO Std. ISO/IEC 8859-3 :1999, Équivalent techniquement à une partie de la norme ECMA 94, plus facile à se procurer.

[ISO8859-4] *Information technology – 8-bit single-byte coded graphic character sets – Part 4 : Latin alphabet No. 4*, ISO Std. ISO/IEC 8859-4 :1998, Équivalent techniquement à une partie de la norme ECMA 94, plus facile à se procurer.

[ISO8859-5] *Information technology – 8-bit single-byte coded graphic character sets – Part 5 : Latin/Cyrillic alphabet*, ISO Std. ISO/IEC 8859-3 :1999.

[ISO8859-6] *Information technology – 8-bit single-byte coded graphic character sets – Part 6 : Latin/Arabic alphabet*, ISO Std. ISO/IEC 8859-6 :1999.

[ISO8859-7] *Information technology – 8-bit single-byte coded graphic character sets – Part 7 : Latin/Greek alphabet*, ISO Std. ISO/IEC 8859-7 :2003.

[ISO8859-8] *Information technology – 8-bit single-byte coded graphic character sets – Part 8 : Latin/Hebrew alphabet*, ISO Std. ISO/IEC 8859-8 :1999.

[ISO8859-9] *Information technology – 8-bit single-byte coded graphic character sets – Part 9 : Latin alphabet No. 5*, ISO Std. ISO/IEC 8859-9 :1999.

[ISO8859-10] *Information technology – 8-bit single-byte coded graphic character sets – Part 10 : Latin alphabet No. 6*, ISO Std. ISO/IEC 8859-10 :1998.

[ISO8859-11] *Information technology – 8-bit single-byte coded graphic character sets – Part 11 : Latin/Thai alphabet*, ISO Std. ISO/IEC 8859-11 :2001.

[ISO8859-13] *Information technology – 8-bit single-byte coded graphic character sets – Part 13 : Latin alphabet No. 7*, ISO Std. ISO/IEC 8859-13 :1998.

[ISO8859-14] *Information technology – 8-bit single-byte coded graphic character sets – Part 14 : Latin alphabet No. 8 (Celtic)*, ISO Std. ISO/IEC 8859-14 :1998.

[ISO8859-15] *Information technology – 8-bit single-byte coded graphic character sets – Part 15 : Latin alphabet No. 9*, ISO Std. ISO/IEC 8859-15 :1999.

[ISO8859-16] *Information technology – 8-bit single-byte coded graphic character sets – Part 15 : Latin alphabet No. 10*, ISO Std. ISO/IEC 8859-16 :2001.

[ISO10646] *Technologies de l'information – Jeu universel de caractères codés sur plusieurs octets (JUC)*, ISO Std. ISO/CEI 10 646 :2003(F). [Online]. Available : http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

[ISO10646a1] *Technologies de l'information – Jeu universel de caractères codés sur plusieurs octets (JUC) – Amendement 1 : Glagolitique, copte, géorgien et autres caractères*, ISO Std. ISO/CEI 10 646 :2003/Amd.1 :2005(F). [Online]. Available : http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

[ISO10646a2] *Technologies de l'information – Jeu universel de caractères codés sur plusieurs octets (JUC) – Amendement 2 : N'Ko, phags-pa, phénicien et autres caractères*, ISO Std. ISOCEI 10 646 :2003/Amd.2 :2006(F). [Online]. Available : http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

[ISO10646a3] *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Amendment 3 : Lepcha, Ol Chiki, Saurashtra, Vai and other characters*, ISO Std. ISO/IEC 10 646 :2003/Amd.3 :2008(E). [Online]. Available : http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

[ISO10646a4] *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Amendment 4 : Cham, Game Tiles, and other characters*, ISO Std. ISO/IEC 10 646 :2003/Amd.4 :2008(E). [Online]. Available : http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

[ISO10646a5] *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Amendment 5 : Tai Tham, Tai Viet, Avestan, Egyptian Hieroglyphs, CJK Unified Ideographs Extension C, and other characters*, ISO Std. ISO/IEC 10 646 :2003/Amd.5 :2008(E). [Online]. Available : http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

[Jona] D. W. Jones. Punched cards. [Online]. Available : http://www.cs.uiowa.edu/~jones/cards/

[Jonb] ——. Punched cards codes. [Online]. Available : http://www.cs.uiowa.edu/~jones/cards/codes.html

[Kat] R. H. Katz. Napoleon's secret weapon. [Online]. Available : http://bnrg.eecs.berkeley.edu/~randy/Courses/CS39C.S97/optical/optical.html

[Kora] J. Korpela. Character histories : notes on some ascii code positions. [Online]. Available : http://www.cs.tut.fi/~jkorpela/latin1/ascii-hist.html

[Korb] ——. A tutorial on character code issues. [Online]. Available : http://www.cs.tut.fi/~jkorpela/chars.html

[Mac80] C. E. Mackenzie, *Coded Character Sets, History and Development*, ser. The systems programming series. Addison-Wesley, 1980.

[Mü] D. Müller. Code de Popham. [Online]. Available : http://www.apprendre-en-ligne.net/crypto/codes/popham.html

[rab] Technical information. [Online]. Available : http://rabbit.eng.miami.edu/info/index.html

[RFC20] V. Cerf. (1969) ASCII format for network interchange. RFC 20. IETF. Probably a copy of X3.4-1968. [Online]. Available : http://www.ietf.org/rfc/rfc20.txt

[RFC821] J. B. Postel. (1982) Simple mail transfer protocol. RFC 821. IETF. [Online]. Available : http://www.ietf.org/rfc/rfc821.txt

[RFC1521] N. Borenstein and N. Freed. (1993) Mime (multipurpose internet mail extensions) part one. RFC 1521. IETF. [Online]. Available : http://www.ietf.org/rfc/rfc1521.txt

[RFC2130] C. Weider, C. Preston, K. Simonsen, H. Alvestrand, R. Atkinson, M. Crispin, and P. Svanberg. (1996) The report of the IAB character set workshop. RFC 2130. IETF. [Online]. Available : http://www.ietf.org/rfc/rfc2130.txt

[Sava] J. J. Savard. The punched card. [Online]. Available : http://www.quadibloc.com/comp/cardint.htm

[Savb] ——. Telecipher devices. [Online]. Available : http://www.quadibloc.com/crypto/tele03.htm

[Sav08] J. Savard. (2008) Re : Vertical bar, broken bar and ASCII code 124. Usenet message on alt.folklore.computers. [Online]. Available : news:a6738911-857a-46a0-a874-f2a9229415f6@d36g2000prf.googlegroups.com

[Sea] S. J. Searle. A brief history of character codes. [Online]. Available : http://tronweb.super-nova.co.jp/characcodehist.html

[Smi64] F. W. Smith, "New american standard code for information interchange," *Western Union Technical Review*, April 1964. [Online]. Available : http://wps.com/projects/codes/New-ASCII/index.html

[Smi67] ——, "Revised U.S.A. standard code for information interchange," *Western Union Technical Review*, November 1967. [Online]. Available : http://wps.com/projects/codes/Revised-ASCII/index.html

[Uni] The unicode consortium home page. Unicode Consortium. [Online]. Available : http://www.unicode.org

[UTR17] K. Whistler, M. Davis, and A. Freytag, "Unicode character encoding model," Unicode Consortium, Tech. Rep. 17, 2008. [Online]. Available : http://www.unicode.org/reports/tr17

[vW] J. van Wingen. Character sets. letters, tokens and codes. [Online]. Available : http://www.terena.org/activities/multiling/euroml/mlcs5.html

[Wika] Polybius square. Wikipedia, The Free Encyclopedia. [Online]. Available : http://en.wikipedia.org/w/index.php?title=Polybius_square&oldid=261174254

[Wikb] Sémaphore (communication). Wikipédia, l'encyclopédie libre. [Online]. Available : http://fr.wikipedia.org/w/index.php?title=S%C3%A9maphore_(communication)&oldid=35629796

[Win] D. T. Winter. Codes. [Online]. Available : http://homepages.cwi.nl/~dik/english/codes