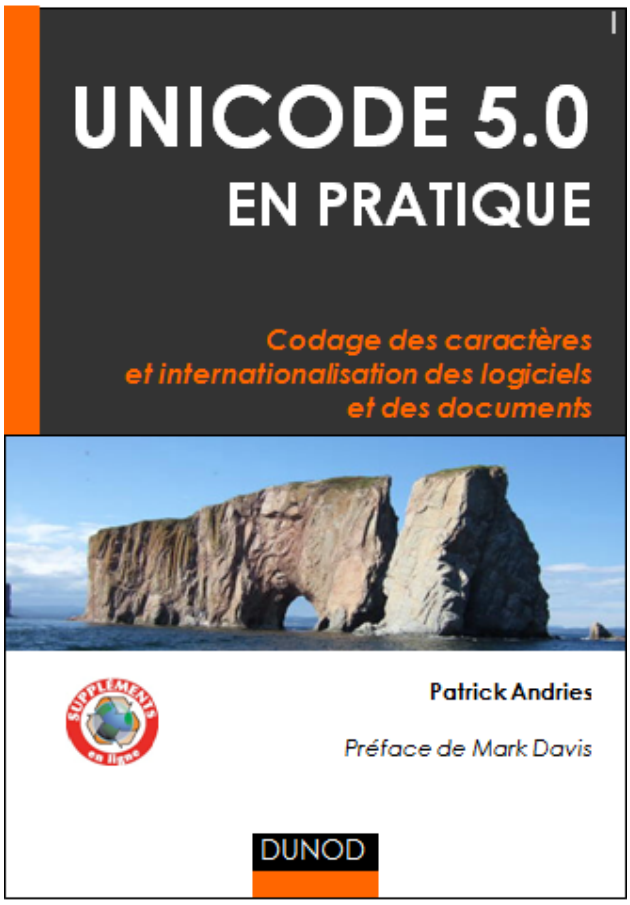
« Je ne connais pas d'autre ressource moderne, ni en anglais ni en français, regroupant une telle gamme de sujets utiles, et les expliquant de façon si claire et si accessible à un large public. J'espère que vous apprécierez cet ouvrage autant que moi. »

Mark Davis, Ph. D.
Président et cofondateur du Consortium Unicode

Commandez en ligne



Mis à jour
22/XII/2003

Introduction à Unicode et à l'ISO 10646

Patrick Andries
membre spécialiste du consortium Unicode

Conseils Hapax
345, rue Sainte-Thérèse
Sault-de-Roxton (Québec)
J0H 1E0
hapax@iquebec.com
http://hapax.iquebec.com

RÉSUMÉ. Cet article présente les points saillants de la version 3.2 du standard Unicode et son pendant ISO, la norme internationale ISO/CEI 10646. L'article répond tout d'abord à quelques questions fréquentes : qu'est-ce qui a poussé au développement de cette norme ? Quels en sont les principes de conception ? Quels caractères normalise-t-elle ? Qu'est-ce qu'un caractère, un glyphe ? Comment doit-on coder les suites de caractères, les caractères accentués ? Suit un bref examen du modèle de codage des caractères et des formes normalisées de représentation des données. Enfin l'article conclut par une description de deux processus fondamentaux et leur rapport avec Unicode : le tri et le rendu.

RÉSUMÉ EN CRI[1] ᐧᐄᓵᐧᐋ ᐋ ᐃᒋᔥᑖᒡ᙮ ᐆ ᐱᔅᑭᑎᓯᓂᐦᐄᑭᓐ, ᐋᐅᒄ ᒑ ᐧᐄᔥᑖᒡ ᐊᓐ ᐋ ᐃᔑᓈᑯᐦᒡ ᐊᓐ ᐹᔨᑯᔨᒡ ᐋ ᐃᔑ ᒥᓯᓂᐦᐄᒑᐱᑎᑭᓅᒡ, 3.2 ᑳ ᐃᑎᓯᓈᑖᒡ, ᑭᔮ ᐊᓐ ᑳ ᐊᔥᒋᐲᒡ ᐹᔨᑯᔨᒡ ᐋᐧᐄ ᐃᔑ ᓂᐦᐄᐲᐦᑖᑭᓅᒡ ᒑᐧᑳᓐ ᒥᓯᐧᐋ ᐅᑖ ᐋ ᐃᔅᐱᑎᔅᑭᒥᑳᒡ ᐊᔅᒌ, ISO/CEI 10646 ᐹᔨᑯᔨᒡ ᑳ ᐃᔑ ᑭᓂᐧᐋᐱᐦᑖᑭᓅᒡ ᒑᐧᑳᓐ᙮ ᐆ ᐱᔅᑭᑎᓯᓂᐦᐄᑭᓐ, ᐋᑯᑦ ᐋ ᓂᔥᐧᑳᔥᑖᒑᒡᐦ ᐊᓂᐦᐄ ᐧᑳᒋᔅᒑᒨᓐᐦ ᒨᔥ ᐋ ᐧᑳᒋᔅᒑᒧᓈᓅᒡ: ᒑᐧᑳᓐ ᐧᐋᐦᒋ ᒋᐦᒋᔥᑖᑭᓅᒡ ᐆ ᑳ ᐃᔑ ᐋᐱᑎᔒᔥᑖᑭᓅᒡ? ᒑᐧᑳᓐ ᒑᔅᒋᓂᐧᐋᐱᐦᑖᑭᓅᒡ ᐊᓐ ᑳ ᐃᔑᓈᑯᐦᑖᑭᓅᒡ? ᑖᓂᐦᐄ ᐧᐋᒋᐦᒡ ᐊᔨᒧᐧᐃᓂᔥᐦ ᒫᔪᐲᐦᑖᑭᓅᒡᐦ? ᒑᐧᑳᓐ ᐊᓐ ᐊᔨᒧᐧᐃᓂᔥ ᐃᔥᐱᔑ ᐧᐄ ᐊᓐ ᒥᔑᓈᐱᔥᑭᐦᐄᑭᓂᐦᑳᓂᔥ? ᑖᓂᑖ ᒑ ᐅᐦᒋ ᓂᐦᐄᐲᒡ ᐋ ᒥᓯᓂᐦᐄᒑᐱᑎᑭᓅᒡ, ᒑᒌ ᐅᐦᒋ ᒋᑭᔥᑖᒡ? ᑖᓐ ᐋ ᐃᔥᒌᐧᔖᒥᑭᐦᒡ ᐊᓐ ᐹᔨᑯᔨᒡ ᒑ ᐃᔑ ᒥᓯᓂᐦᐄᒑᐱᑭᓅᒡ ᒨᔥ ᒑ ᒋᔅᒋᓄᐧᐋᐱᐦᑖᑭᓅᒡ? ᐊᐱᔒᔥ ᐋ ᐊᔨᒧᑎᑭᓅᒡ ᐊᓐ ᒑ ᐃᔑ ᐧᐄᔥᑖᑭᓅᒡ ᓂᓈᐦᑰ ᐊᔨᒧᐧᐃᓂᔥᐦ ᑖᓂᑦ ᒑ ᐃᐦᑎᑯᐦᒡ ᐅᐦᒋ ᒥᓯᓂᐦᐄᒑᐲᒡᐦ, ᐊᓐ ᑭᔮ ᐊᔨᒧᐧᐃᓂᔥ ᒑ ᐃᔑ ᐧᐄᔥᑖᑭᓅᒡ ᒑ ᒋᔅᒋᓄᐧᐋᐱᐦᑖᑭᓅᒡ ᑭᔮ ᐊᓐ ᑯᐃᔅᒄ ᒑ ᐃᔑ ᐧᐄᔥᑖᒡ᙮ ᐋᐅᒄ ᒦᓐ ᒑᒃ ᒑ ᑭᓂᐧᐋᐱᐦᑖᑭᓅᒡ ᓂᓈᐦᑰ ᐋ ᐃᔑ ᒥᓯᓂᐦᐄᒑᓅᒡ (ᐱᑯᓂᒑ ᐋ ᐃᔨᔫᔥᑖᒡ ᐋ ᐋᐱᑎᒡ ᐊᓂᑦ ᐹᔨᑯᔨᒡ ᐋ ᐃᔑ ᒥᓯᓂᐦᐄᒑᐲᒡ) ᒑᒃ ᒦᓐ ᒑ ᐊᑎ ᐧᐄᔥᑖᒡ ᓃᔓ ᒑᐧᑳᓂᐦᐃ ᐅᐦᒋ ᐊᓐ ᐋ ᒥᓯᓂᐦᐄᒑᐱᑎᑭᓅᒡ ᐹᔨᑯᔨᒡ ᑳ ᐃᔑᓈᑯᐦᑖᑭᓅᒡ: ᐋᐅᒄ ᑯᐃᔅᒄ ᐋ ᐱᒋᔅᑎᓂᑭᓅᒡ ᑭᔮ ᑖᓂᑖ ᒑ ᐃᔑᓈᑯᐦᒡ᙮

---

[1] L'écriture des Cris est prise en charge par le bloc *syllabaire autochtone canadien* d'Unicode (U+1400..U+167F). Il s'agit d'un syllabaire original utilisé par la majorité des langues arctiques et subarctiques d'Amérique septentrionale. La variante crie reproduite ci-dessus est parlée au Nord du Québec. Les Cris forment un des rameaux de la famille algonquienne ou algique (parmi ses proches parents, on trouve au Québec les Montagnais, les Têtes-de-boule et les Naskapis). Près de 150 000 Cris vivent en Amérique du Nord.

Document numérique. Volume 6 – nº 3-4/2002, pages 51 à 88

ABSTRACT. This article presents a concise overview of Unicode 3.2 and its ISO equivalent, ISO/IEC 10646. It introduces and summarizes the main features and design principles before examining the character encoding model, normalizations forms and common processes such as sorting and rendering and their relationship with Unicode.

MOTS-CLÉS : Unicode, ISO/CEI 10646, formes normalisées, modèle de caractères, principes de conception, écritures, alphabets, glyphe, caractère, tri, rendu, polices, OpenType.

ᐊᓕᒍ·ᐃᓐᓇᖅ ᐊᑦ ᑭᖅᑲᓕᖅᑕᐅᓚᖅ: ᐸᓕᔪᓕᐊᓂ ᐊᑦ ᐃᓂ ᒥᔪᓂᖅᐃᓚᐱᑎᓕᐊᓂᒃ; ISO/CEI 10646, ᔪᐃᕐᓇ ᐊᑏ ᐃᓂ ᒥᔪᓂᖅᐃᓚᐱᒃ, ᐃᓕᓴᐅ, ᐊᑏ ᑭᐱᓇ·ᐊᐱᖅᑕᐸᓂᒃ ᐊᑦ ᐅᓪᑕᐸᓂᒃ ᐊᑏ ᐃᓂᔨᔪᖅᑐᑦ, ᒥᔪᓂᖅᐃᓚᐅᓐᖅ, ᒥᓂᔨᐱᓪᓚᐸᓂᖅᐃᓚᐳᓂᖅ, ᐊᓕᒍ·ᐃᓂᓪᑐᖅ, ᔪᐃᕐᓇ ᐊᑦ ᐱᑭᐱᓂᐸᓂᒃ, ᑕᓂᑕ ᐊᑏ ᐃᓂᔨᔪᖅᑐᑦ, ᓂᔨᖅᑯ ᐊᑦ ᒥᔪᓂᖅᐃᓚᐱᑎ, ᑲ ᐊᐱᖅᐸᓪᓚᑦ ᒥᔪᓂᖅᐃᓚᓱᔨᑕᑦ.

KEYWORDS: Unicode, ISO/IEC 10646, normalisation forms, character model, design principles, glyphs, collation, rendering, fonts, OpenType.

1. Introduction

1.1. Pourquoi Unicode ?

Les jeux de caractères utilisés avant l'avènement d'Unicode possédaient des architectures très différentes les uns des autres[2]. Pour plusieurs, la simple détection des octets représentant un caractère était un processus contextuel complexe. Les jeux de caractères classiques ne pouvaient au mieux prendre en charge que quelques langues. La prise en charge de plusieurs langues à la fois était difficile, voire impossible. Aucun jeu de caractères ne fournissait toutes les lettres, les signes de ponctuation et les symboles techniques en usage courant utilisés pour une seule langue comme le français.

Produire des logiciels destinés à différents marchés se résumait à une alternative simple : écrire des versions différentes du logiciel pour chaque jeu de caractères ou rendre le code du logiciel nettement plus compliqué pour traiter tous les jeux de caractères nécessaires. Ces deux options impliquent plus de codage, plus de tests, plus d'entretien et plus de soutien aux utilisateurs. Elles rendent la production de logiciels destinés à l'exportation plus chère et retardent leur lancement ce qui n'est pas sans conséquence sur les revenus.

Le marché actuel rassemble souvent des données provenant de différentes sources (il suffit de penser à l'Internet). Le caractère codé correspondant à une lettre aussi simple que « A » peut varier en fonction du jeu de caractères et rendre le repérage, le tri et d'autres opérations fort difficiles. D'autres problèmes surgissent quand on mélange des données provenant de différents jeux de caractères (comme c'est le cas lors d'un copier-coller entre deux fichiers codés différemment). Il faut alors étiqueter ou baliser les données avec des informations sur le jeu de caractères d'origine et créer de la sorte un nouveau codage à états. Perpétuer les anciennes

2. On trouvera dans l'article de Jacques André dans ce numéro une introduction historique aux diverses normes de codage de caractères.

méthodes de production dans ce contexte ne peut qu'entraîner des coûts prohibitifs ou, pire, conduire au désastre.

Le standard Unicode[3,4] est un mécanisme universel de codage de caractères. Il définit une manière cohérente de coder des textes multilingues et facilite l'échange de données textuelles. Obligatoire pour la plupart des nouveaux protocoles de l'Internet[5], mis en œuvre dans tous les systèmes d'exploitation et langages informatiques modernes[6], Unicode est la base de tout logiciel qui veut fonctionner aux quatre coins du monde.

Grâce à Unicode, l'industrie informatique peut assurer la pérennité des données textuelles tout en évitant la prolifération de jeux de caractères et en augmentant l'interopérabilité des données. Enfin, Unicode simplifie le développement de logiciels et en réduit les coûts. En effet, Unicode permet de coder tous les caractères utilisés par toutes les langues écrites du monde (plus d'un million de caractères sont réservés à cet effet). Tous les caractères, quelle que soit la langue dans laquelle ils sont utilisés, sont accessibles sans aucune séquence d'échappement. Le codage de caractère Unicode traite les caractères alphabétiques, les caractères idéographiques et les symboles de manière équivalente, avec comme conséquence qu'ils peuvent se côtoyer dans n'importe quel ordre avec une égale facilité.

Le standard Unicode attribue à chacun de ses caractères un numéro[7] et un nom. À ce titre, il ne diffère guère des autres standards ou normes de codage de caractères. Cependant Unicode fournit d'autres renseignements cruciaux afin de s'assurer que le texte codé sera lisible : la casse des caractères codés, leur directionalité et leurs propriétés alphabétiques. Unicode définit également des renseignements sémantiques et comprend des tableaux de correspondance de casse ou des conversions entre Unicode et les répertoires de divers autres jeux de caractères importants.

À l'heure actuelle, les données Unicode peuvent être codées sous trois formes principales : une forme codée sur 32 bits (UTF-32), une forme sur 16 bits (UTF-16) et une forme de 8 bits (UTF-8) conçue pour faciliter son utilisation sur les systèmes ASCII préexistants. Le standard Unicode est identique à la norme internationale ISO/CEI 10646 en ce qui concerne l'affectation des caractères (leur numéro) et leurs

3. Généralement toute mention au « standard Unicode » s'applique également à la norme internationale ISO/CEI 10646.

4. Une traduction française complète annotée et mise à jour du Standard Unicode est disponible à l'adresse <http://hapax.iquebec.com>. Cette traduction a bénéficié du concours financier du gouvernement du Québec.

5. XML, HTML, WML, Corba 3.0, LDAP, etc.

6. Exemples : Java, ECMAScript (Javascript), MS Windows 2000 et Xp, Mac OS/X.

7. Une « valeur scalaire Unicode », dans le jargon d'Unicode.

noms[8]. Toute application qui se conforme à Unicode se conforme donc à l'ISO/CEI 10646.

1.2. Principes directeurs

L'objectif principal du standard Unicode était de remédier à deux écueils sérieux et fréquents dans la plupart des programmes informatiques multilingues : l'ambiguïté des polices qui utilisent souvent les mêmes valeurs pour coder des caractères et des symboles totalement différents et l'utilisation de multiples jeux de caractères incompatibles provenant de normes nationales et industrielles contradictoires. Ainsi, même en Europe occidentale, des conflits existent entre la page de code 850 et l'ISO/CEI 8859-1. Dans le cas de logiciels qui prennent en charge les idéogrammes d'Extrême-Orient, le même ensemble d'octets utilisés en ASCII peut également servir de deuxième octet pour les caractères à deux octets. Ces logiciels doivent donc être capables de distinguer les caractères ASCII codés sur un octet des caractères à deux octets.

Afin de résoudre ces problèmes, le consortium Unicode a conçu une méthode uniforme de codage des caractères à la fois plus efficace et plus souple que les systèmes de codage dits historiques. Le nouveau système devait satisfaire les besoins techniques multilingues tout en codant un grand éventail de caractères qui permettent d'assurer les besoins typographiques de qualité professionnelle et de la micro-édition dans le monde entier.

1.3. La norme ISO/CEI 10646 et le standard Unicode

En 1984, le JTC1/SC2 de l'ISO/CEI[9] définissait le mandat du groupe de travail numéro 2 (GT2) : « élaborer une norme établissant un répertoire de caractères graphiques des langues écrites du monde et son codage ». À la même époque, Joe Becker de Xerox à Palo Alto travaillait sur un jeu de caractères universel qu'il nommait Unicode. Quelques années plus tard, en 1988, plusieurs industriels, dont Xerox, se réunirent pour former ce qui allait devenir le consortium Unicode. Conscients des bénéfices d'une seule norme universelle de codage de caractères, les membres du Consortium Unicode collaborèrent avec les représentants de l'Organisation internationale de normalisation (ISO) à la fin 1991.

À la suite de négociations officielles sur la convergence des deux codes, les deux répertoires fusionnèrent en janvier 1992. Depuis lors, une étroite collaboration et une liaison officielle entre les deux comités ont permis d'assurer la synchronisation des amendements et additions aux deux normes de sorte que leurs répertoires respectifs et leurs codages sont aujourd'hui identiques.

---

8. Le standard Unicode ne précise pas de noms français, cependant la norme internationale ISO/CEI 10646 a été publiée en anglais et en français. Les noms utilisés ici sont les noms officiels de la version française de l'ISO/CEI 10646.

9. On trouvera à la fin de ce numéro un inventaire des divers abréviations ou sigles utilisés.

Le répertoire du standard Unicode est rigoureusement identique à celui de l'ISO/CEI 10646. Cette identité point par point s'applique à tous les caractères codés des deux normes y compris les caractères idéographiques extrême-orientaux (également appelés caractères han). La norme ISO/CEI 10646 attribue à chaque caractère un nom et une valeur de code, le standard Unicode assigne les mêmes noms et valeurs de code mais il fournit également d'importants algorithmes de mise en œuvre, des propriétés de caractères et d'autres renseignements sur la sémantique de ceux-ci.

Le comité technique du consortium Unicode (CTU[10]) est le groupe de travail au sein du consortium chargé de la création, de la mise à jour et de la qualité du standard Unicode. Le CTU contrôle toutes les données techniques fournies au consortium et prend des décisions quant au contenu de ces données. Les membres de plein droit du consortium, qui doivent s'acquitter d'une cotisation annuelle, se prononcent sur les décisions du CTU. Les membres associés, les membres experts ainsi que les dirigeants du consortium font partie du CTU, mais ne bénéficient pas du droit de vote. Le président du CTU peut inviter d'autres personnes à participer aux discussions techniques, sans droit de vote toutefois.

La dernière version papier d'Unicode est la version 3.0, publiée sous la forme d'un épais livre en 2000. Elle a été mise à jour par deux fois à l'aide de rapports séparés de sorte que la version actuelle d'Unicode est la version 3.2. La prochaine version papier d'Unicode, la 4.0, est prévue pour l'automne 2003.

Unicode 4.0 est bien paru en septembre 2003

2. Quels caractères Unicode et l'ISO/CEI 10646 normalisent-ils ?

La base de tout jeu de caractères est une correspondance entre des valeurs de code (des numéros) et des caractères. Il ne faut pas confondre un caractère avec sa représentation visuelle (le glyphe représentatif qui l'illustre dans les tableaux de caractères). C'est pourquoi Unicode fournit les renseignements sémantiques nécessaires pour décrire les caractères qu'il code. Nous reviendrons sur cette distinction fondamentale entre caractère et glyphe par la suite (v. 3.1, Caractère abstrait, caractère codé et glyphe).

L'espace de codage du standard Unicode se divise en plans de 64K cellules (voir la figure 1). Chaque cellule peut se voir affecter un caractère. Le premier plan (plan 00_{16}), le plan multilingue de base (PMB), comprend des caractères usuels dans les écritures alphabétiques, syllabiques et idéographiques ainsi que divers chiffres et symboles. Le contenu des zones à usage privé du PMB et des plans à usage privé ($0F_{16}$ et 10_{16}) n'est pas prescrit par Unicode, en d'autres mots on est libre d'y affecter des caractères de son choix. La zone d'indirection est un ensemble de cellules réservées pour UTF-16 (voir la figure 2 et la section 6.3, Forme en mémoire

10. Voir l'entretien dans ce même numéro avec le directeur du comité technique du consortium Unicode, Ken Whistler.

des caractères) afin de pouvoir distinguer les caractères codés sur un seizet de ceux codés sur deux. La figure 3 donne le détail des premiers blocs du PMB.

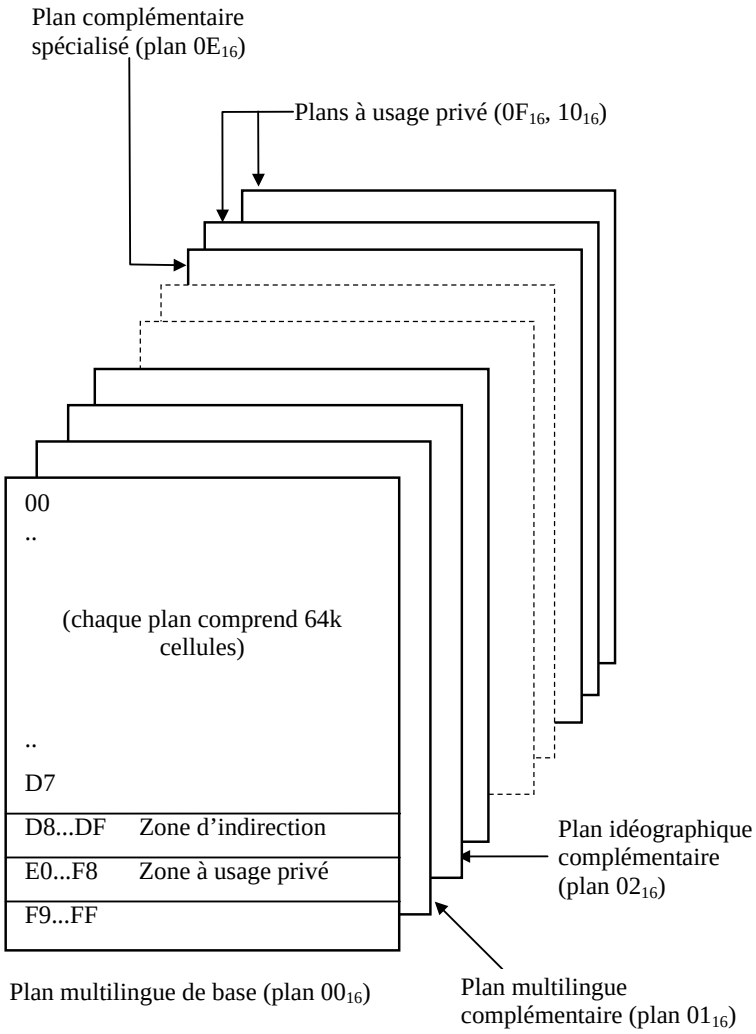
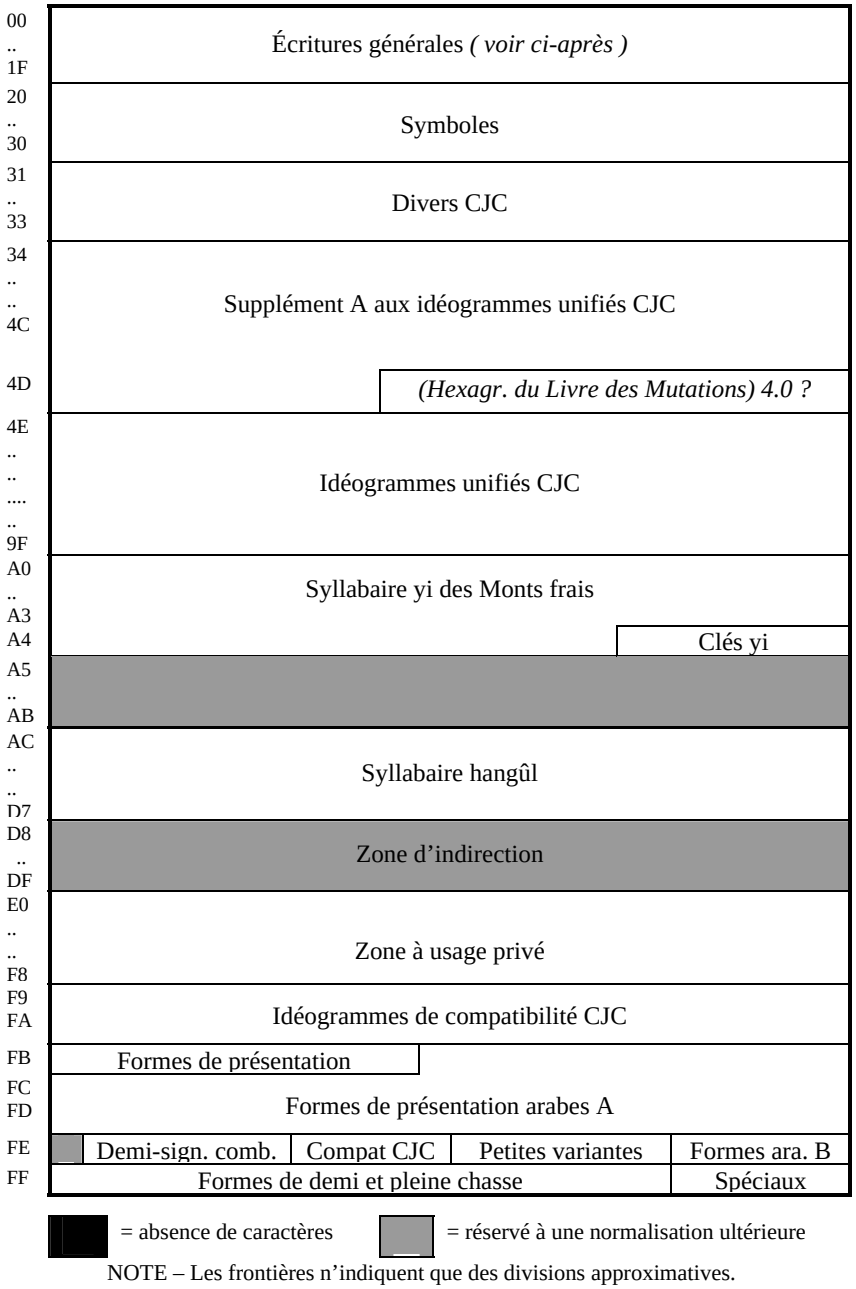


Figure 1. *Plans et zones de codage*



Le nom de ce bloc a changé: il s'agit désormais du Classique des Mutations.

NOTE – Les frontières n'indiquent que des divisions approximatives.

Figure 2. *Plan multilingue de base (PMB)*

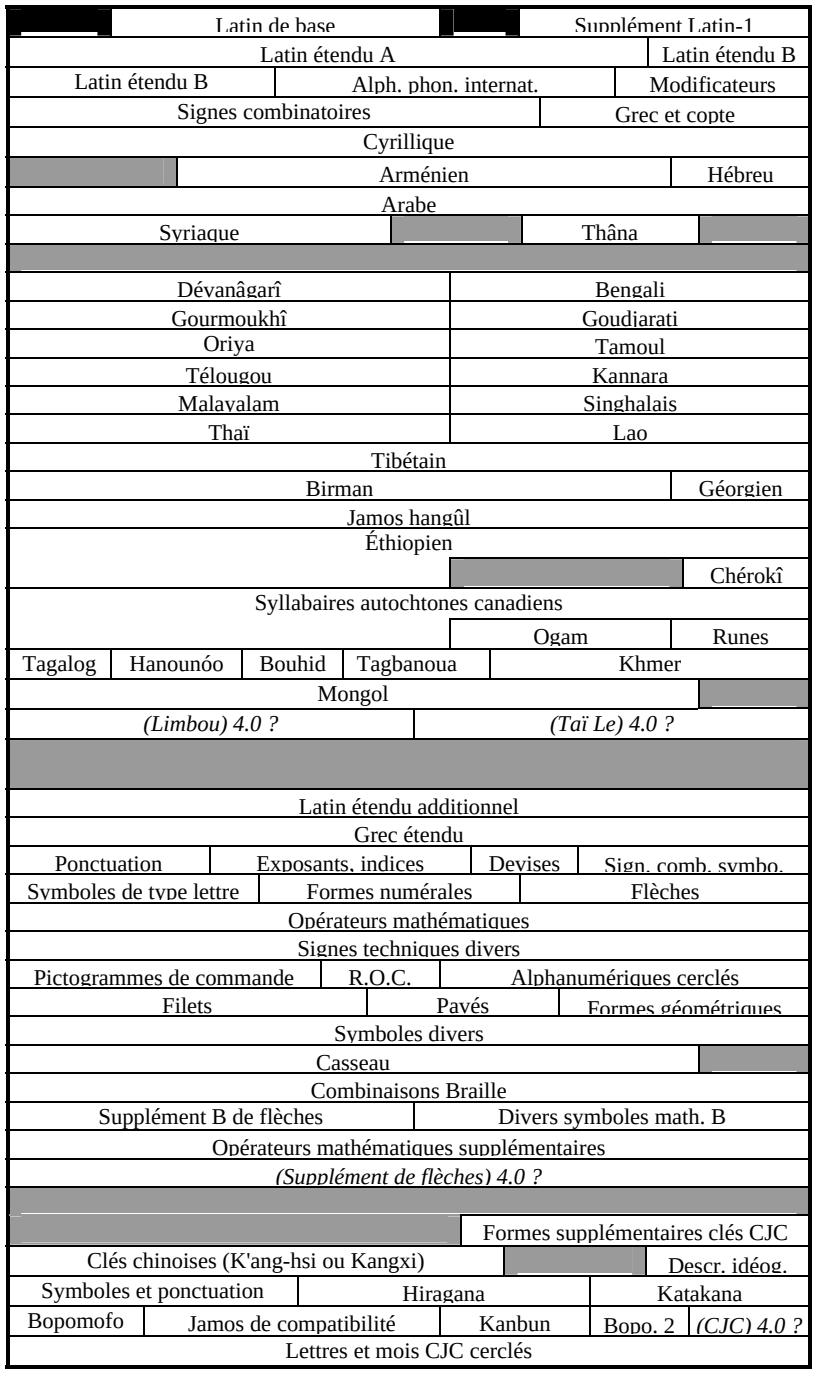


Figure 3. *Zone des écritures générales du PMB*

Les noms de des blocs de la rangée 19 ont changé, il s'agit désormais du limbu et du taï-le.

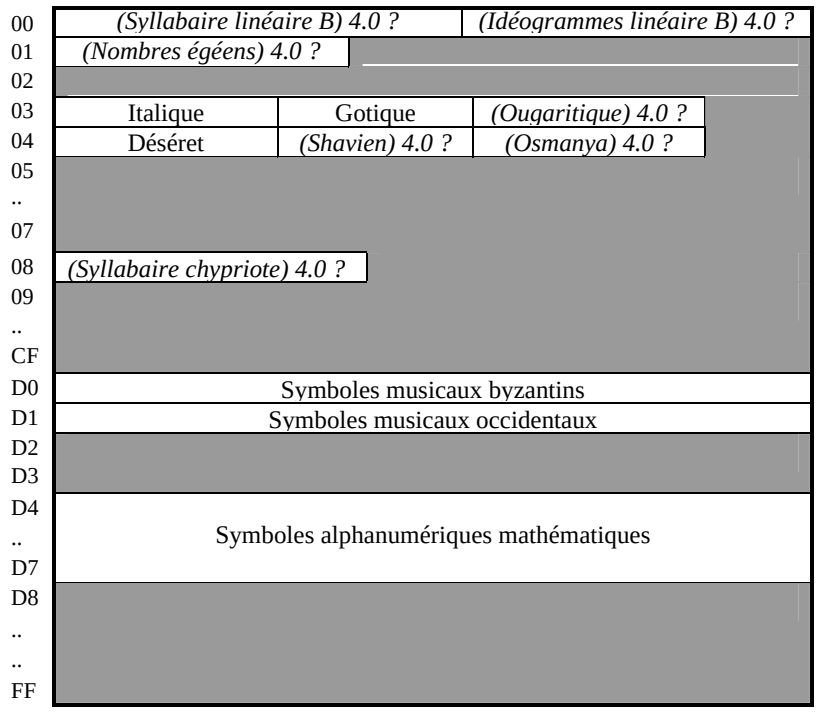


Figure 4. *Plan multilingue complémentaire (PMC)*

Le plan multilingue complémentaire (plan 01_{16}, voir la figure 4) est destiné aux systèmes d'écritures non idéographiques et aux symboles similaires à ceux que l'on retrouve dans le PMB, mais contrairement au PMB la plupart des écritures qui y sont codées (et qui y seront codées) ne sont plus utilisées de nos jours. Le Plan idéographique complémentaire (plan 02_{16}) normalise un ensemble de caractères utilisés en Extrême-Orient. Le Plan complémentaire spécialisé (plan $0E_{16}$) normalise un ensemble de caractères graphiques à vocation particulière (certains de ces caractères n'ont pas d'apparence visuelle). Les noms en italique qui figurent dans ces figures indiquent des blocs d'écriture qui devraient faire partie de la prochaine version d'Unicode, la version 4.0.

Tous ces blocs ont bien été inclus dans Unicode 4.0.

La dernière version d'Unicode, la version 3.2, code actuellement 95 221 caractères, la grande majorité dans le PMB. Il reste environ 6 700 positions de code non affectées dans le PMB, elles sont réservées à une future normalisation. Ajoutons que 870 000 positions de code restent disponibles dans les autres plans.

3. Caractères

3.1. Caractère abstrait, caractère codé et glyphe

Un des défis les plus redoutables dans la conception d'un jeu de caractères universel réside dans la conception souvent différente que se fait chaque langue de ce qui constitue un élément textuel fondamental (une lettre dans le cas des écritures alphabétiques). Ce concept peut même changer, pour une même langue, selon les applications textuelles considérées. Ainsi, en typographie traditionnelle allemande, la combinaison de lettres « ck » constitue un élément textuel pour ce qui a trait à la coupure de mots (où cette combinaison se transforme en « k-k ») mais pas en ce qui concerne le tri lexicographique. En français, les objets « a » et « A » sont habituellement considérés comme des éléments textuels distincts pour ce qui a trait au rendu, mais on les confond fréquemment, à dessein, lors de recherches textuelles.

Une norme de codage de caractères fournit des unités fondamentales de codage (que l'on nomme caractères abstraits) mises en correspondance biunivoque (de un à un) avec le numéro qu'on leur attribue. Ces numéros de code sont les plus petites unités adressables du texte stocké.

Le graphème constitue une des classes d'éléments textuels importantes, il correspond typiquement à ce que l'utilisateur considère être un caractère. La figure 5 illustre le rapport qui existe entre les caractères abstraits et les graphèmes.

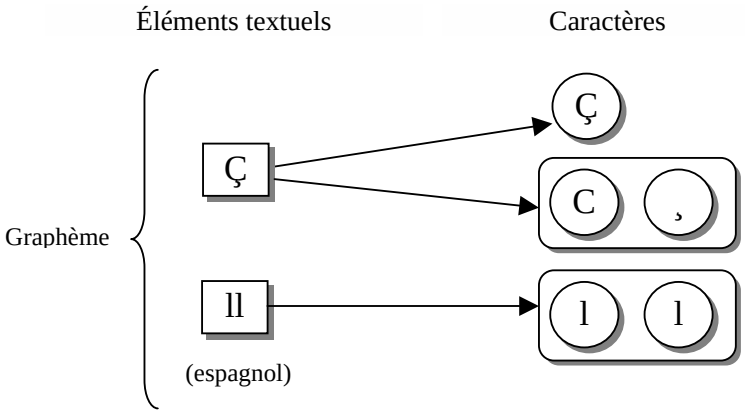


Figure 5. *Éléments textuels et caractères*

Comment passe-t-on des éléments textuels ou des graphèmes aux caractères abstraits d'un jeu de caractères ? Il faut comprendre qu'aucune architecture de jeu de caractères ne peut satisfaire de manière optimale tous les processus textuels (rendu, coupure de lignes, détection de mots, correction orthographique, indexation, synthèse vocale, sélection et mise en évidence à l'écran, édition, comparaison, tri,

compression, etc.). L'ensemble des caractères abstraits choisis forme donc un compromis entre les besoins des différents traitements textuels pour un ensemble de langues utilisant la même écriture. En effet, pour des raisons d'efficacité et d'échange de données, Unicode et l'ISO/CEI 10646 ne codent pas nécessairement les graphèmes d'une langue particulière mais des caractères abstraits correspondant aux caractères communs à une écriture (pouvant servir à plusieurs langues).

En résumé, on dira qu'un *caractère abstrait* est une unité d'information utilisée pour organiser, commander ou représenter des données textuelles. Le caractère abstrait n'a pas de forme concrète (il ne faut pas le confondre avec ses glyphes) et ne correspond pas nécessairement à ce que l'utilisateur imagine être un « caractère » dans sa langue (un graphème). Le glyphe[11] représente les différentes formes qu'un caractère abstrait peut prendre (voir la figure 6). Un *caractère codé*, pour sa part, est l'association entre un caractère abstrait et son numéro. Notons que certains caractères abstraits sont codés plus d'une fois dans Unicode[12].

Glyphes (œils)	Caractères Unicode/ ISO 10646
A 𝒜 𝔸 𝘈 A A	U+0041 lettre majuscule latine a[13]
ffi 𝑓𝑓𝑖 ffi ffi	U+0066 lettre minuscule latine f + U+0066 lettre minuscule latine f + U+0069 lettre minuscule latine i
ف ف	U+0647 lettre arabe fa'

Figure 6. *Caractères et glyphes*

3.2. Caractères combinatoires et diacritiques

On nomme caractères combinatoires les caractères destinés à s'afficher en association avec un caractère de base. Si l'on décide de représenter « ö » à l'aide d'un « o » (U+004F) suivi d'un caractère « ¨ » (U+0302), « o » est le caractère de

11. On oppose parfois *l'image de glyphe*, dessin concret et invariant (sauf, sans que ce soit explicite ni toujours vrai, pour un facteur d'échelle) au *glyphe* plus abstrait, qui fait abstraction des variations typographiques (autre que le facteur d'échelle) mais pas des variantes de forme contextuelles comme en arabe. Quand on voudra éviter la polysémie du mot glyphe, on utilisera le terme traditionnel typographique *œil* (pluriel des œils) pour désigner sans équivoque l'image de glyphe. La lettre arabe sîn (un caractère abstrait) peut être représentée, selon le contexte, par un de ses quatre glyphes archétypiques, eux-mêmes représentés concrètement par des œils fruits du travail soigné d'un fondeur.

12. C'est le cas des équivalents de compatibilité avec d'autres jeux de caractères ou des rares doublons (dans le cas du coréen, ils sont reproduits sciemment à des fins de compatibilité).

13. On représente le numéro des caractères Unicode à l'aide de la notation U+*xxxx* (où *xxxx* est la valeur hexadécimale d'un caractère codé du PMB) ou U+*yyyyyy* (où *yyyyyy* est la valeur hexadécimale d'un caractère codé dans un plan complémentaire). La forme à cinq chiffres (U+*zzzzz*) semble exclue par Unicode.

base et « ¨ » est un caractère combinatoire. Pour bien indiquer qu'il s'agit d'un caractère combinatoire, on le représente surmontant un cercle pointillé : ◌̈. Lors du rendu, les glyphes qui représentent ces caractères seront dessinés dans un certain ordre près du glyphe qui représente le caractère de base. Le standard Unicode distingue deux types de signes combinatoires : ceux avec chasse et ceux sans. Les caractères combinatoires sans chasse (encore appelés à chasse nulle) n'occupent pas de place par eux-mêmes. Toutefois, la combinaison du caractère de base et du caractère à chasse nulle peut occuper au rendu plus d'espace latéral que le caractère de base seul. Ainsi, un « î » chasse légèrement plus qu'un simple « i ».

Les diacritiques constituent la principale classe des caractères combinatoires sans chasse utilisés dans les alphabets européens. Dans le standard Unicode, la définition du terme « diacritique » est relativement vague et comprend à la fois les accents ainsi que d'autres signes sans chasse. Tout caractère de base peut se voir adjoindre n'importe quel diacritique quelle que soit l'écriture de ce caractère. On peut donc théoriquement adjoindre un « ^ » à une lettre arabe.

3.3. Suite de caractères de base et diacritiques

Tous les caractères combinatoires doivent suivre le caractère de base qu'ils qualifient. La suite de caractères Unicode U+0075 LETTRE MINUSCULE LATINE U + U+0308 ◌̈ DIACRITIQUE TREMA + U+0065 LETTRE MINUSCULE LATINE E représente sans équivoque « üe » et non « uë ». La convention d'ordonnancement utilisée par le standard Unicode correspond à l'ordre logique des diacritiques arabes et indiens dont la grande majorité suit (au clavier et dans la chaîne parlée) les caractères de base par rapport auxquels ils se placent.

Dans les tableaux consacrés aux écritures de l'Inde, certaines voyelles sont représentées à la gauche d'un cercle pointillé dans les tableaux de caractères (cf. figure 7). Il faut distinguer attentivement ce cas spécial de celui des diacritiques habituels. Ces signes-voyelles doivent être rendus à la gauche de la consonne ou du groupe consonantique même si logiquement (en mémoire) ils suivent la consonne dans leur représentation Unicode. On dit que ces signes-voyelles sont antéposés.

Figure 7. *Signe vocalique indien antéposé (i bref)*

3.4. Caractères combinatoires multiples

Il arrive que plusieurs diacritiques s'adjoignent à un seul caractère (cf. figure 8). Le standard Unicode ne limite pas le nombre de caractères combinatoires qui peut suivre un caractère de base.

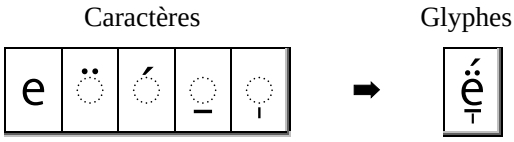


Figure 8. *Empilement de diacritiques*

Si des caractères combinatoires peuvent affecter une même zone typographique — par exemple un U+0308 ◌̈ DIACRITIQUE TRÉMA et un U+0301 ◌́ DIACRITIQUE ACCENT AIGU — l'ordre des caractères codés détermine l'ordre d'affichage (cf. figure 9). Les diacritiques sont disposés à tour de rôle à partir du glyphe correspondant au caractère de base pour ensuite s'en éloigner. Les caractères combinatoires placés au-dessus d'un caractère de base s'empilent verticalement, en commençant par le premier dans l'ordre de stockage logique. On poursuit l'empilement vers le haut d'autant de signes qu'il est nécessaire selon le nombre de codes de caractère qui suivent le caractère de base. La situation est inversée pour les caractères combinatoires placés sous un caractère de base : les caractères combinatoires s'empilent vers le bas. On appelle cet ordre l'ordre centrifuge.

ê	lettre minuscule e accent circonflexe lettre minuscule e + accent circonflexe
ė	lettre minuscule e + diacritique point en chef
ệ	lettre minuscule e accent circonflexe + diacritique point souscrit lettre minuscule e + accent circonflexe + diacritique point souscrit lettre minuscule e + diacritique point souscrit + accent circonflexe
ẹ̇	lettre minuscule e + diacritique point en chef + diacritique point souscrit lettre minuscule e + diacritique point souscrit + diacritique point en chef
ë́	lettre minuscule e tréma + diacritique accent aigu lettre minuscule e + diacritique tréma + diacritique accent aigu
é̈	lettre minuscule e accent aigu + diacritique tréma lettre minuscule e + diacritique accent aigu + diacritique tréma

Figure 9. *Interaction des caractères diacritiques*

Certains caractères dérogent à l'empilement centrifuge. C'est le cas du grec, lorsqu'un esprit précède un accent aigu ou grave, les deux diacritiques doivent alors s'écrire côte à côte au-dessus de la lettre de base. Ici l'ordre des caractères est *code du caractère de base + code de l'esprit + code de l'accent.* Cet exemple démontre

bien que la disposition correcte des signes diacritiques dépend de l'écriture employée.

4. Principes de conception du standard Unicode

Le standard Unicode énumère dix principes qui ont guidé son élaboration. Pour un jeu de caractères aussi vaste qu'Unicode, tous ces principes ne s'appliquent pas toujours de manière égale. Parfois, l'adhésion à un principe entraîne la violation d'un des autres principes. Unicode est le résultat d'un délicat équilibre entre différents besoins parfois contradictoires.

UNIVERSALITÉ — Le standard Unicode fournit un répertoire unique et universel.

EFFICACITÉ — L'analyse et le traitement d'un texte Unicode sont simples.

CARACTÈRES ET NON GLYPHES — Le standard Unicode définit des caractères et non des glyphes.

SÉMANTIQUE — Les caractères ont une sémantique bien définie.

TEXTE BRUT — Le standard Unicode normalise du texte brut.

ORDRE LOGIQUE — La représentation implicite en mémoire correspond à l'ordre logique.

UNIFICATION — Le standard Unicode unifie les caractères identiques à l'intérieur des écritures et non à l'intérieur d'un même alphabet associé à une seule langue.

COMPOSITION DYNAMIQUE — On compose les formes à diacritique(s) dynamiquement.

SÉQUENCE ÉQUIVALENTE — À chaque forme statique précomposée correspond au moins une suite équivalente de caractères décomposés que l'on pourra recomposer dynamiquement.

CONVERTIBILITÉ — Unicode définit des conversions exactes entre son répertoire et d'autres normes largement répandues.

4.1. Universalité[14]

Le standard Unicode décrit un seul jeu de caractères de très grande taille qui comprend tous les caractères nécessaires aux écritures du monde. Le passage d'une écriture à l'autre ne nécessite pas, contrairement à certains jeux de caractères d'Extrême-Orient, de caractère d'échappement. Les caractères de toutes les écritures peuvent se mêler sans aucun encombre (voir la figure 10).

Glyphe représentatif	Numéro	Nom officiel français de l'ISO/CEI 10646
D	U+0044	LETTRE MAJUSCULE LATINE D
v	U+0076	LETTRE MINUSCULE LATINE V
o	U+006F	LETTRE MINUSCULE LATINE O
ř	U+0159	LETTRE MINUSCULE LATINE R CARON
a	U+0061	LETTRE MINUSCULE LATINE A
k	U+006B	LETTRE MINUSCULE LATINE K
	U+0020	ESPACE
天	U+5929	IDÉOGRAMME UNIFIÉ CJC-5929
心	U+2F3C	CLÉ CHINOISE CŒUR
س	U+0633	LETTRE ARABE SÎN
ل	U+0644	LETTRE ARABE LAM
ا	U+0627	LETTRE ARABE ALIF
م	U+0645	LETTRE ARABE MÎM
α	U+03B1	LETTRE MINUSCULE GRECQUE ALPHA
𐌸	U+10338	LETTRE GOTIQUE TH
𝅗𝅥	U+1D110	SYMBOLE MUSICAL POINT D'ORGUE

14. Ce principe remplacera dans la version 4.0 le premier principe d'Unicode encore officiel dans la version actuelle d'Unicode : « Unicode est un code à 16 bits ». Ce principe est caduc depuis l'introduction d'UTF-32. Unicode peut se représenter de manière aussi légitime sous trois formes stockées différentes (UTF-8, UTF-16 ou UTF-32).

≮	U+2270	NI PLUS PETIT NI ÉGAL À
ཿ	U+0F06	SIGNE TIBÉTAIN D'INSERTION YIG MGO PHUR SHAD MA[15]

Figure 10. *Universalité du jeu de caractères*

4.2. Efficacité

Afin de permettre des mises en œuvre efficaces, Unicode n'inclut pas de caractères d'échappement ou d'états de passage. Tous les caractères Unicode ont la même qualité, tous les codes sont d'une égale facilité d'accès[16]. Par convention et dans la mesure du possible, les caractères d'une même écriture sont regroupés en un bloc de caractères consécutifs. Non seulement ceci facilite-t-il la consultation des tableaux de caractères, mais cela permet également des mises en œuvre plus compactes. La majorité des caractères de ponctuation communs aux différentes écritures sont regroupés dans un bloc séparé.

Cette universalité et cette uniformité de codage permettent une analyse, un tri, un affichage, un repérage et une édition efficaces des chaînes textuelles Unicode.

4.3. Caractères et non glyphes[17]

Le standard Unicode distingue les caractères, qui forment les plus petites unités distinctives et significatives d'une langue écrite, des glyphes qui représentent les différentes formes visuelles qu'un caractère peut prendre. Un seul glyphe peut représenter un seul ou plusieurs caractères (on parle alors d'une ligature). De même, de multiples glyphes peuvent représenter un seul caractère (**b**, *b* et ***b*** sont tous des U+0062 b).

On représente un caractère par un numéro qui ne réside qu'en mémoire ou sur disque. Le standard Unicode ne s'intéresse qu'au codage de caractères et non aux glyphes. Contrairement aux caractères, les glyphes apparaissent à l'écran ou sur papier et représentent une forme particulière d'un ou plusieurs caractères. Un répertoire de glyphes constitue une police. La forme des glyphes et les méthodes d'identification et de sélection de glyphes sont la responsabilité des fonderies et des normes appropriées. Celles-ci ne font pas partie du standard Unicode (v. 9, Rendu).

Pour certaines écritures, comme l'arabe et différentes écritures du sous-continent indien, le nombre de glyphes nécessaires à l'affichage peut être nettement plus

15. Il s'agit d'une translittération (« étymologique ») d'origine anglaise du tibétain, la transcription française (et donc sa prononciation approximative) est fournie par son synonyme non normatif défini par l'ISO/CEI 10646 : *yik go p'our ché ma*.

16. Comme on le verra par la suite (v. 6, Modèle de codage des caractères), ceci dépend de la forme stockée choisie. Ainsi les caractères stockés à l'aide d'UTF-8 et UTF-16 ne sont-ils pas aussi faciles à accéder que les caractères stockés sous la forme UTF-32.

17. Voir également les articles de Yannis Haralambous et d'Olivier Randier dans ce même numéro.

important que le nombre des caractères qui codent les unités de base de cette écriture. Le nombre de glyphes peut également dépendre du style calligraphique de la police. Ainsi, une police arabe de style nastaliq peut comprendre plusieurs milliers de glyphes. Toutefois, quelle que soit la police choisie, un texte arabe est toujours représenté par la même cinquantaine de caractères codés (la même cinquantaine de numéros).

4.4. Sémantique

Les caractères sont munis d'une sémantique bien définie. Le standard Unicode fournit des tableaux de propriétés de caractère que l'on peut utiliser pour le tri, l'analyse ou d'autres algorithmes qui ont besoin d'une connaissance sémantique des caractères traités. Voici quelques-unes des propriétés de caractère définies par Unicode :

- la casse[18] du caractère : majuscule 𝒜, minuscule a, casse de titre Dz ;
- sa directionalité : gauche, droite, faible, neutre ;
- s'il est à symétrie miroir : < ∫ ⟶ (à ajuster selon directionalité du texte) ;
- s'il s'agit d'un numéral : *iv* 7 ;
- s'il est combinatoire : ◌̇ ◌̈ f◌̂ ◌̰ .

4.5. Texte brut

Un *texte brut* est constitué d'une suite de codes de caractères, dont aucun ne représente du balisage. Un texte brut Unicode est donc une suite simple de codes de caractères Unicode. À l'inverse, un *texte enrichi*, également connu sous le nom de *texte de fantaisie*, est constitué de texte brut et d'un ensemble d'informations comme la force de corps employée, la couleur du texte, des hyperliens, etc. Le texte de cet article, par exemple, est un texte formaté à l'aide de plusieurs polices, il s'agit d'un texte enrichi.

La simplicité du texte brut lui permet de servir de base aux textes enrichis. SGML, HTML, XML ou TEX sont des textes enrichis représentés à l'aide de textes bruts émaillés de suites de caractères qui représentent des structures de données supplémentaires (ici des balises). On définit parfois le texte brut de la façon suivante : un texte brut doit comprendre suffisamment d'information pour que le texte puisse être affiché de manière lisible, sans plus. Enfin, notons que, si le texte brut est public, normalisé et lisible par tous, la représentation du texte enrichi est souvent propriétaire ou peut varier selon les mises en œuvre.

18. Casse, dans ce sens générique, est un néologisme puisque la casse se référait uniquement à l'origine à la boîte contenant tous les caractères d'imprimerie. On parle, toutefois, depuis longtemps de *bas de casse* pour désigner une minuscule (v. Le Robert).

4.6. Ordre logique

Les textes Unicode, quelle que soit l'écriture, sont stockés en mémoire en ordre logique. Cet ordre correspond *grosso modo* à l'ordre dans lequel le texte est saisi au clavier. Parfois, l'ordre des caractères à l'affichage ou à l'impression diffère de l'ordre logique, c'est le cas des écritures bidirectionnelles comme l'arabe ou l'hébreu. La figure 11 illustre la différence entre les ordres logique et d'affichage.

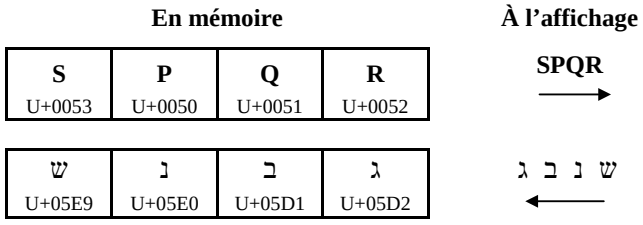


Figure 11. *Ordre logique*

À l'affichage, le glyphe qui représente le premier caractère du texte latin de la figure 11 s'affiche à gauche. En revanche, le premier caractère logique du texte hébreu est représenté par le glyphe hébreu le plus proche de la marge droite. Les glyphes hébreux suivants sont disposés vers la gauche.

On utilise l'ordre logique même lorsque des caractères de différentes directions dominantes se mêlent : écritures gauche-à-droite (grec, cyrillique, latin), droite-à-gauche (arabe, hébreu) ou même verticale. Les propriétés de directionalité implicites aux caractères suffisent généralement à déterminer l'ordre correct d'affichage. Toutefois, il arrive parfois que cette directionalité implicite s'avère insuffisante pour afficher le texte de manière lisible. Unicode prévoit des caractères de commande qui permettent d'influencer l'ordre d'affichage.

Rappelons que certains caractères, comme l'*i bref* dévanâgarî (ि), s'affichent avant les caractères qu'ils suivent logiquement en mémoire. Les signes combinatoires (comme les accents en français, en grec, en cyrillique, les points voyelles arabes, les signes voyelles dévanâgarî, etc.) n'apparaissent pas de manière linéaire dans le texte rendu. Dans une chaîne de caractères Unicode, tous ces signes *suivent* le caractère de base qu'ils modifient, contrairement à ce qui fait généralement avec les machines à écrire. Ainsi, un « b̈ » latin est-il stocké sous la forme d'un « b » suivi du signe combinatoire (ou diacritique) « ◌̈ ».

4.7. Unification

Le standard Unicode évite de coder plusieurs fois le même caractère et n'attribue qu'un seul numéro de code à un caractère d'une écriture donnée, quel que soit le nombre de langues qui utilise ce caractère (le *a* de *chat*, *Katze*, *gato* et *kat* est donc codé à l'aide d'un seul numéro : U+0061). Il en va de même pour les idéogrammes chinois/japonais/coréens (CJC).

Le standard Unicode ne tente pas de coder des caractéristiques de texte comme la langue, la police, la force de corps, l'emplacement, l'œil des caractères (glyphes), etc. Ainsi, ne conserve-t-il pas d'information sur la langue lors du codage de caractères : l'*i grec* français, l'*ypsilon* allemand et le *wye* anglais sont tous représentés par le même code de caractère, U+0057 « Y », il en va de même pour le *zi* (*tseu*) chinois, le *ji* japonais et le *ja* coréen représentés par le même code de caractère, U+5B57 字, car ils partagent origine et sens et ne connaissent que des différences d'œil anodines.

Toutefois un autre principe (voir 4.10, Convertibilité) s'oppose parfois à l'unification des caractères. Ainsi, si un des jeux de caractères fondateurs d'Unicode distingue deux caractères d'apparence similaire ou identique, Unicode maintient cette distinction et code deux fois le caractère. C'est le cas en latin pour le U+00C5 Å et U+212B Å. C'est également le cas pour les caractères han (劍, 劍, 劒, 劒, 劒, 釼) qui désignent tous l'épée ou le poignard et sont de structure identique mais qui sont codés séparément dans le jeu de caractères japonais source JIS X 0208-1990 ; Unicode les distingue donc pour assurer une conversion aller-retour sans perte vers ce jeu de caractères.

4.8. Composition dynamique

Le standard Unicode permet de composer dynamiquement les formes accentuées des lettres et des syllabes hangûl. Ce processus est extrêmement productif, car il est ouvert. On peut donc créer de nouvelles formes à l'aide d'une combinaison de caractères de base[19] suivis d'un ou plusieurs diacritiques. Ainsi, le tréma « ◌̈ » peut-il s'adjoindre à des voyelles ou à des consonnes dans des langues qui utilisent l'écriture latine ainsi que dans d'autres écritures. Dans les écritures comme l'arabe ou l'hébreu, les diacritiques représentent souvent des voyelles. Les écritures de l'Inde comportent de nombreux caractères combinatoires dont les glyphes suivent habituellement, à la même hauteur sur la ligne, le caractère de base.

4.9. Séquence équivalente

Certains éléments textuels peuvent être codés de plus d'une façon : une forme précomposée ou une ou plusieurs compositions dynamiques. Pour des raisons de compatibilité avec les jeux de caractères actuels, on a inclus dans le standard de nombreuses formes précomposées. Les caractères précomposés sont équivalents à leur suite de caractères décomposée. Ainsi « ä » est-il équivalent à « a » + « ◌̈ ». Unicode fournit pour chaque forme précomposée incluse dans la norme son équivalent canonique décomposé. Ces suites sont fournies dans la base de données Unicode et dans les tableaux de caractères (v. 5.5, Décompositions).

19. Par exemple, dans le cas des diacritiques doubles qui recouvrent les deux caractères de base qui précèdent.

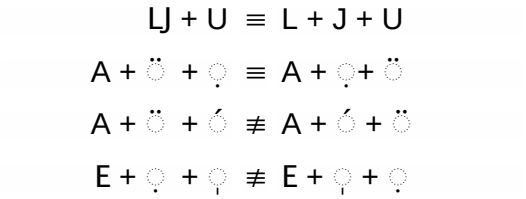


Figure 12. *Séquences équivalentes et distinctes*

Des suites de diacritiques qui interagissent au niveau typographique (par exemple, deux accents suscrits ou deux diacritiques souscrits) sont considérées distinctes si les diacritiques se présentent dans des ordres différents. Unicode attribue pour chaque caractère une classe combinatoire et définit, à l'aide ces classes combinatoires, un algorithme de mise en ordre canonique des suites de signes combinatoires. Cet ordre canonique permet de trier les séquences équivalentes de caractères combinatoires dans un ordre prévisible. Les caractères de base sont de classe combinatoire zéro. Tous les diacritiques qui interagissent l'un sur l'autre au niveau typographique sont de même classe combinatoire, des suites d'ordre différent les contenant ne sont donc pas équivalentes.

	Original	**Décomposition**	**Ordre canonique**
Chaîne	Ä + ̣	➡ A + ¨ + ̣	➡ A + ̣ + ¨
Classe combi.	0 220	0 230 220	0 220 230
Chaîne	A + ̣ + ¨	➡	➡ A + ̣ + ¨
Classe combi.	0 220 230		0 220 230

Figure 13. *Classes combinatoires et mise en ordre canonique*

4.10. Convertibilité

Le standard Unicode et l'ISO/CEI 10646 formaient à leur naissance un jeu de caractères complètement neuf. Il a donc fallu définir des méthodes d'importation et d'exportation de données codées dans des jeux de caractères fondamentaux à l'époque[20]. Un grand soin a été apporté afin de garantir une conversion aller-retour sans perte entre Unicode et ces jeux de caractères. C'est pourquoi, lorsque des

20. Unicode ne prévoit de conversions que dans le cas de jeux de caractères élaborés avant mai 1993.

variantes de forme (ou même la même forme) se sont vues affectées des numéros de caractère différents dans une norme de base, Unicode les garde séparées.

En général, toute valeur de code dans une autre norme correspond à une valeur de code en Unicode. Il arrive, cependant, qu'une valeur de code dans une autre norme correspondra à une suite de valeur de code Unicode ou vice-versa.

5. Le standard Unicode : mode d'emploi

Le standard Unicode est un ouvrage monumental, il répertorie en effet plus de 95 000 caractères et définit les règles qui les régissent. Cet inventaire planétaire, fruit de plus d'une décennie de travail international, se présente sous la forme d'une longue liste. Cette liste de 600 pages est annotée et elle est précédée de 13 chapitres. Les premiers chapitres sont normatifs et décrivent des aspects communs à toutes les écritures (et résumés dans cet article), ils sont suivis de chapitres « introductifs » essentiels qui portent, chacun, sur une famille d'écritures particulières.

Les tableaux du standard Unicode représentent les caractères de la norme ISO/CEI 10646 et du standard Unicode. Les caractères sont regroupés en blocs apparentés. Habituellement, ces blocs comprennent des caractères provenant d'une seule écriture. Un bloc de caractères suffit généralement à représenter une écriture. Il existe, toutefois, des exceptions notables, plus particulièrement dans le domaine des caractères de ponctuation.

Une liste de noms de caractère suit chaque tableau de codes[21], à l'exception des idéophonogrammes CJC et des syllabes hangûl dont un algorithme définit les noms. La liste de noms de caractère reprend tous les caractères du bloc et fournit le plus souvent des renseignements complémentaires sur ceux-ci.

5.1. Liste des noms de caractère

L'exemple suivant illustre les différentes parties de chaque entrée de la liste de noms de caractère.

Code	*Glyphe*	*Description*	
00AE	®	SYMBOLE MARQUE DÉPOSÉE	*(nom ISO 10646)*
00AF	¯	MACRON = barre supérieure APL • ce caractère chasse → 02c9 ˉ lettre modificative macron → 0304 ◌̄ diacritique macron → 0305 ◌̅ diacritique tiret haut	*(nom ISO 10646)* *(nom optionnel)* *(renseignement)* *(renvois)*

21. La liste intégrale des noms de caractère est disponible sous forme de texte brut à l'adresse suivante : <http://iquebec.ifrance.com/hapax/ListeDesNoms.htm>

≈ 0020 , 0304 ◌̄ *(décomposition de compatibilité)*

00E5 å LETTRE MINUSCULE LATINE A ROND EN CHEF
- danois, norvégien, suédois, wallon

≡ 0061 a 030A ◌̊ *(décomposition canonique)*

5.2. Images dans les tableaux de codes et dans les listes de caractères

Chaque caractère dans ces tableaux de codes est représenté à l'aide d'un glyphe représentatif. Ce glyphe n'a pas de valeur normative, il permet simplement à un utilisateur averti de reconnaître le caractère visé ou de retrouver facilement le caractère dans les tableaux. Dans de nombreux cas, il existe des représentations concurrentes, plus ou moins établies, pour un même caractère.

Les concepteurs de polices de haute qualité devront effectuer leur propre recherche pour déterminer l'apparence la plus appropriée des caractères Unicode. En outre, de nombreuses écritures nécessitent des formes de glyphes différentes selon le contexte, un positionnement contextuel du glyphe ou la formation de ligatures. Aucune de ces informations n'est illustrée dans les tableaux de codes.

Pour les écritures non européennes, on a choisi comme styles de caractère ceux dont les œils se différenciaient le plus les uns des autres.

5.3. Renvois

Les caractères de renvoi (précédés de →) désignent plusieurs types de renvoi : l'inégalité explicite, les autres casses du même caractère ou d'autres rapports linguistiques.

Inégalité explicite. Les deux caractères ne sont pas identiques bien que leurs glyphes soient identiques ou fort similaires.

003A : DEUX-POINTS
→ 0589 : point arménien
→ 2236 : rapport

Autres rapports linguistiques. Parmi ces rapports, on compte les translittérations (entre le serbe et le croate, par exemple), des caractères sans rapport typographique mais qui servent à représenter le même son, etc.

01C9 lj LETTRE MINUSCULE LATINE LJ
= digramme soudé lj
→ 0459 љ lettre minuscule cyrillique lié
≈ 006C l 006A j

5.4. Renseignements sur les langues

Quand cela peut-être utile, on trouve parfois une note informative identifiant les langues qui utilisent ce caractère. Pour les écritures bicamérales, cette information

n'est fournie que pour les lettres de bas de casse afin d'éviter une répétition inutile. Les points de suspension «…» indiquent que la liste des langues n'est pas limitative et qu'elle ne reprend que les langues principales.

5.5. Décompositions

La séquence de décomposition d'un caractère (constituée d'une ou plusieurs lettres) peut être de deux types : canonique ou de compatibilité. La correspondance canonique est indiquée à l'aide d'un symbole *identique à* ≡.

00E5 å LETTRE MINUSCULE LATINE A ROND EN CHEF
- danois, norvégien, suédois, wallon

≡ 0061 a 030A ◌̊

212B Å SYMBOLE ANGSTRÖM
≡ 00C5 Å lettre majuscule latine a rond en chef

Les correspondances ne fournissent pas des décompositions complètes, c'est l'application récursive des correspondances qui fournit la décomposition canonique (après sa mise en ordre canonique). Voir 7, Formes normalisées.

1EA5 ấ LETTRE MINUSCULE LATINE A ACCENT CIRCONFLEXE ET ACCENT AIGU
≡ 00E2 â 0301 ◌́

00E2 â LETTRE MINUSCULE LATINE A ACCENT CIRCONFLEXE
≡ 0061 a 0302 ◌̂

La correspondance de compatibilité s'indique à l'aide d'un signe *presque égal à* ≈ . Une balise de formatage peut accompagner la décomposition.

01F3 dz LETTRE MINUSCULE LATINE DZ
≈ 0064 d 007A z

FF25 A LETTRE MAJUSCULE LATINE E PLEINE CHASSE
≈ <large> 0045 E

6. Modèle de codage des caractères

Dans la discussion qui précède, nous avons considéré les caractères abstraits et leur numérotation sans jamais nous pencher sur la forme que prendront ces numéros de caractères lors d'un stockage ou d'un transfert. Unicode définit un modèle de codage de caractères qui définit cinq niveaux de représentation des caractères :

– RÉPERTOIRE DE CARACTÈRES ABSTRAITS — ensemble des caractères à coder, c'est-à-dire un alphabet ou un jeu de symboles ;

– JEU DE CARACTÈRES CODÉS — fonction qui attribue un numéro (ou exceptionnellement deux) à chaque caractère abstrait du répertoire ;

- FORME EN MÉMOIRE DES CARACTÈRES — relation entre un ensemble de numéros de caractères utilisés dans un jeu de caractères codés et un ensemble de suites d'unités de stockage (par exemple des octets ou des seizets) ;
- MÉCANISME DE SÉRIALISATION DE CARACTÈRES — correspondance entre des unités de stockage et des suites d'octets sérialisés ;
- SURCODAGE DE TRANSFERT — transformation réversible[22] des données codées, celles-ci peuvent contenir du balisage.

La figure 14 illustre le rapport qui existe en les quatre premiers niveaux du modèle. Les flèches pleines relient les caractères codés (également appelés caractères numérotés) aux caractères abstraits qu'ils représentent et codent. La flèche creuse part d'une suite de caractères codés qui représente sous forme décomposée le caractère abstrait, aucun de ces deux caractères codés ne correspond donc directement au caractère abstrait. Enfin, les rangées *stocké* et *sérialisé* représentent différentes formes de stockage et de sérialisation de caractères codés, nous les décrirons plus en détail ci-après.

6.1. Répertoire de caractères abstraits

Un répertoire est un ensemble de caractères abstraits destinés à être codés, il s'agit habituellement d'un alphabet connu ou d'un ensemble de symboles. Le mot abstrait signifie simplement que ces objets sont définis par convention, comme le sont les 26 lettres de l'alphabet latin, les majuscules ou les minuscules. Il est bon de rappeler que le répertoire comprend des caractères, et non des glyphes. Un répertoire est un ensemble non ordonné.

Contrairement à la plupart des jeux de caractères existants dont le répertoire est fermé (l'ajout d'un nouveau caractère entraîne la définition d'un nouveau jeu de caractère), le répertoire d'Unicode est ouvert. Unicode se veut un codage universel, tout caractère abstrait qui pourra jamais être codé est donc un membre potentiel du jeu à coder, que l'on connaisse actuellement ce caractère ou non.

22. Une conversion vers un autre jeu de caractères ou une compression de textes. Unicode définit une méthode de compression de données appelées SCSU.

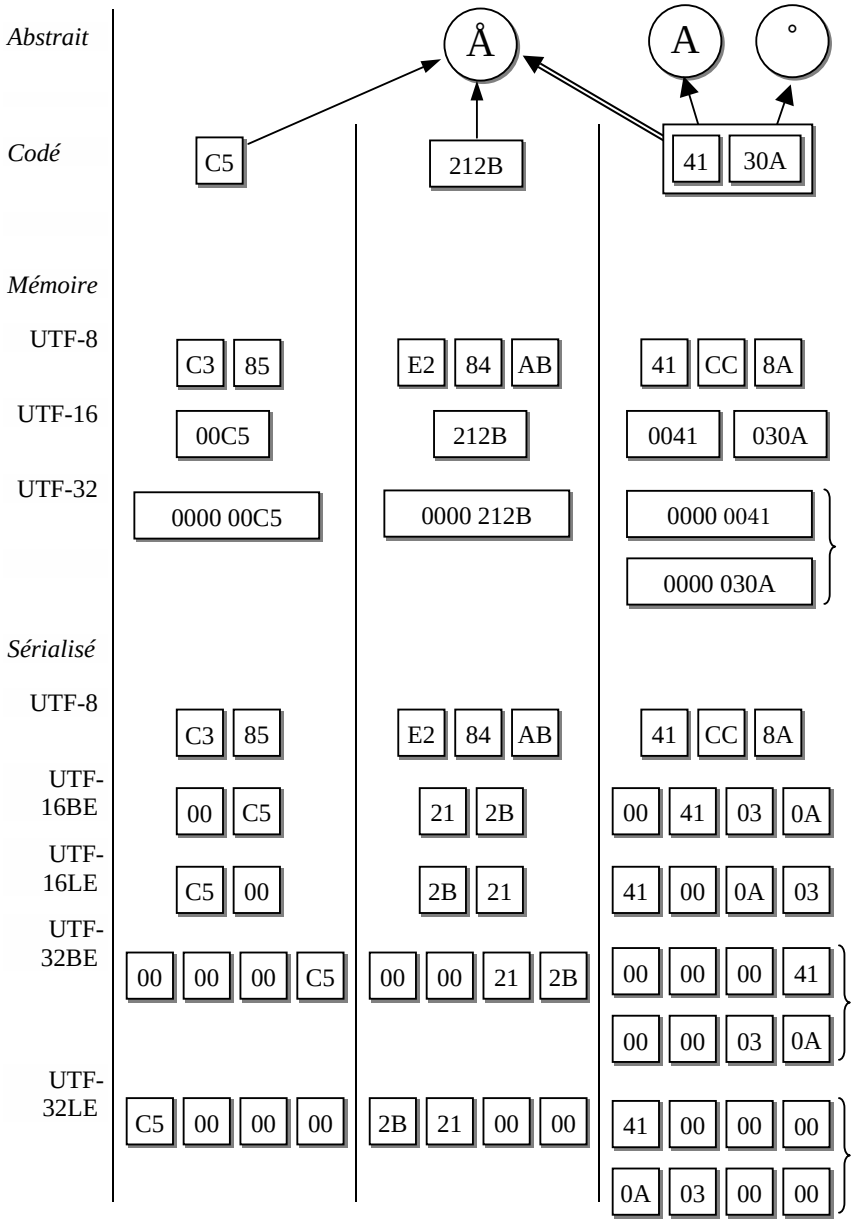


Figure 14. *Modèle de codage des caractères*

6.2. Jeu de caractères codés

On définit un jeu de caractères codés comme la correspondance entre un ensemble de caractères abstraits et un ensemble d'entiers non négatifs. Ce dernier ensemble peut ne pas être contigu. On dit qu'un caractère abstrait est codé dans un jeu de caractères donné si un numéro de caractère existe pour ce caractère.

6.3. Forme en mémoire des caractères

Une forme en mémoire des caractères (ou la forme « naturelle » des caractères) définit la relation entre un ensemble de numéros de caractères utilisés dans un jeu de caractères codés et un ensemble de suites d'unités de stockage en mémoire. Une unité de stockage en mémoire est un entier d'une certaine largeur (exemples : octet ou seizet) qui sert d'unité de base à l'expression des numéros de caractère dans la mémoire d'un ordinateur. Cette forme en mémoire définit la représentation réelle des caractères codés comme données informatiques. Les suites d'unités de stockage en mémoire ne doivent pas nécessairement toutes avoir la même longueur.

Dans les jeux de caractères traditionnels, il n'existe le plus souvent qu'une seule façon de représenter les caractères en mémoire. La *forme en mémoire* est alors inhérente (et implicite) au jeu de caractères codés. En revanche, Unicode et l'ISO/CEI 10646 définissent plusieurs formes « naturelles » de mémorisation. On distingue les formes à largeur fixe des formes à largeur variable.

Formes en mémoire à largeur fixe :

– 16 bits (UCS-2) — chaque numéro de caractère est représenté par un seizet ; cette forme n'existe qu'en ISO/CEI 10646 ; elle ne permet que d'adresser les caractères du PMB.
– 32 bits (UCS-4, UTF-32) — chaque numéro de caractère est représenté par une quantité sur 32 bits. L'espace de code est arbitrairement limité à 0..10FFFF pour des raisons de compatibilité avec UTF-16 (voir ci-après).

Formes en mémoire à largeur variable :

– UTF-8 — suite d'une à quatre unités de mémoire (des octets ici) pour Unicode et théoriquement d'une à six unités de mémoire pour l'ISO/CEI 10646 (en pratique l'ISO/CEI 10646 utilise également jusqu'à quatre unités puisque désormais la limite supérieure de l'espace de codage de l'ISO/CEI est également $10FFFF_{16}$).
– UTF-16 — suite d'une à deux unités de mémoire (des seizets ici).

La forme en mémoire de caractères constitue un des aspects fondamentaux de l'internationalisation des logiciels : combien d'unités utilise-t-on en mémoire pour représenter chaque caractère ? On exprime cette relation à l'aide de deux paramètres : la largeur des unités de mémoire (8, 16 ou 32 bits) et le nombre d'unités de mémoire utilisé pour représenter chaque caractère.

UTF-16

Cette forme en mémoire du standard Unicode est à 16 bits : l'unité de stockage est le seizet. Les caractères du PMB sont codés avec un seul seizet, ceux des autres plans sont codés à l'aide de deux seizets (dits d'indirection). Chaque paire d'indirection est composée d'un seizet d'indirection supérieur (dont la valeur appartient à l'intervalle D800..DBFF[23]) et d'un seizet d'indirection inférieur (DC00..DFFF). La figure 15 illustre le passage du numéro de caractère à la forme en mémoire UTF-16.

UTF-8

Afin de satisfaire les besoins des systèmes architecturés autour de l'ASCII ou d'autres jeux de caractères à un octet, le standard Unicode définit une forme en mémoire supplémentaire : l'UTF-8. Il s'agit d'une forme en mémoire à largeur variable transparente pour les systèmes ASCII.

Il s'agit d'une forme de mémorisation très fréquemment adoptée pour effectuer la transition de systèmes existants vers Unicode. Elle a notamment été choisie comme forme préférée pour l'internationalisation des protocoles d'Internet.

UTF-8 est un codage constitué de suites d'octets de longueur variable ; les bits de poids le plus fort d'un octet indiquent la position de celui-ci dans la suite d'octets. La figure 15 montre comment les bits d'un numéro de caractère Unicode sont répartis parmi les octets UTF-8. Toute suite d'octets qui ne suit pas ce modèle est une suite mal formée d'octets. La forme en mémoire UTF-8 assure la transparence de toutes les valeurs de code ASCII (0..127). En outre, ces mêmes valeurs n'apparaissent pas dans les octets transformés sauf en tant que représentation directe de ces mêmes valeurs ASCII. Les caractères de l'intervalle ASCII (U+0000..U+007F) sont stockés à l'aide d'un seul octet, la plupart des caractères issus des alphabets ou syllabaires sont stockés à l'aide de deux octets, les autres valeurs inférieures à $FFFF_{16}$ avec trois octets et, enfin, les valeurs de 10000_{16} à $10FFFF_{16}$ à l'aide de quatre octets.

Autres caractéristiques importantes de l'UTF-8 :

- Conversion efficace à partir de ou vers un texte codé en UTF-16 ou en UTF-32.
- Le premier octet indique le nombre d'octets, ceci permet une analyse rapide du texte vers l'avant.
- Recherche rapide du début de tout caractère, quel que soit l'octet où l'on commence la recherche dans un flux de données, il suffit de consulter au plus quatre octets en amont pour reconnaître aisément le premier octet qui code le caractère.

23. Sans *U+*, car ce ne sont pas des numéros de caractère mais des valeurs d'unité de stockage

– UTF-8 est un mécanisme de stockage relativement compact en termes d'octets.

– Un tri binaire des chaînes UTF-8 donne le même résultat qu'un tri binaire effectué sur les numéros de caractères (les valeurs scalaires) Unicode correspondantes.

UTF-32

Dans certains cas, on pourra préférer l'utilisation d'une forme en mémoire sur 32 bits, où chaque numéro de caractère Unicode (également appelé valeur scalaire Unicode) correspond à une unité codée sur 32 bits. Même les applications qui n'utilisent pas cette forme voudront sans doute convertir des données en mémoire à partir de et vers ce format à des fins d'interopérabilité.

Quelques caractéristiques importantes de cette forme en mémoire :

– À des fins de compatibilité avec UTF-16, UTF-32 est restreint aux valeurs appartenant à l'intervalle 0..10FFFF$_{16}$ qui correspond parfaitement au domaine des caractères défini par le standard Unicode et l'ISO/CEI 10646.

Numéro du caractère	**UTF-16**	**UTF-8**			
		1er octet	**2e octet**	**3e octet**	**4e octet**
00000000 0*xxxxxxx*	00000000 0*xxxxxxx*	0*xxxxxxx*			
00000*yyy yyxxxxxx*	00000*yyy yyxxxxxx*	110*yyyyy*	10*xxxxxx*		
zzzzyyyy yyxxxxxx	*zzzzyyyy yyxxxxxx*	1110*zzzz*	10*yyyyyy*	10*xxxxxx*	
000*uuuuu zzzzyyyy yyxxxxxx*	110110*ww wwzzzzyy*[a] + 110111*yy yyxxxxxx*	11110*uuu*	10*uuzzzz*	10*yyyyyy*	10*xxxxxx*

Figure 15. *Distribution binaire des octets en UTF-16 et UTF-8*

a. Où *wwww* = *uuuuu* – 1

– Le standard Unicode ajoute un certain nombre de contraintes de conformité, en sus de celles précisées par l'ISO/CEI 10646, plus particulièrement quant à la sémantique des caractères. Déclarer un texte UTF-32 plutôt qu'UCS-4 permet de spécifier l'application de ces règles sémantiques propres à Unicode.

– De par sa forme de mémorisation à largeur fixe (sur 32 bits), la valeur numérique d'un caractère Unicode en UTF-32 correspond toujours à son numéro de caractère codé.

6.4. Mécanisme de sérialisation de caractères

Un mécanisme de sérialisation définit une correspondance entre des unités en mémoire et des suites d'octets sérialisés. Cette transformation permet de définir des formes d'échange de données persistantes intermachine dont les unités de stockage (en mémoire) sont plus grandes que l'octet. Il se peut alors qu'une permutation des octets s'avère nécessaire afin d'ordonner les données selon la polarité canonique des octets adoptée par la plate-forme en question.

En particulier :

– – UTF-16, dont les unités de base sont des seizets, doit préciser l'ordre des octets à adopter lors de la sérialisation. C'est là la différence entre UTF-16BE, dont les deux octets de seizet sont sérialisés en ordre gros-boutien[24], et l'UTF-16LE[25] qui les sérialisent en ordre petit-boutien.

– UTF-32, dont les unités de base sont des quantités à 32 bits, doit préciser l'ordre des octets à adopter lors de la sérialisation. Tout comme pour l'UTF-16, il existe donc deux mécanismes de sérialisation : UTF32-BE sérialise les octets en ordre gros-boutien alors qu'UTF-32LE les sérialise en ordre petit-boutien. UTF32-BE est structurellement identique à l'UCS-4 tel que défini par l'ISO/CEI 10646-1:2000.

Il est important de ne pas confondre forme en mémoire de caractères et mécanisme de sérialisation de caractères :

– La *forme en mémoire de caractères* fait correspondre des unités de mémorisation aux numéros de caractère, alors que le *mécanisme de sérialisation* de caractères fait correspondre ces unités de mémorisation à des octets sérialisés destinés à être échangées avec d'autres systèmes.

– Le *mécanisme de sérialisation* doit préciser de quelle façon les unités de mémorisation sont transformées en suites d'octets.

---

24. Architecture informatique où l'octet de poids fort des valeurs numériques multioctets est stocké en premier. L'image est empruntée aux « Voyages de Gulliver », dans lequel les Gros-Boutiens, partisans de la théorie selon laquelle il faut casser un œuf par le gros bout, s'opposent aux Petits-Boutiens qui, eux, soutiennent qu'il faut, au contraire, casser un œuf par le petit bout.

25. BE et LE renvoient respectivement aux initiales de la traduction anglaise de grand-boutien et de petit-boutien.

7. Formes normalisées

Afin de garantir une compatibilité aller-retour entre les normes préexistantes et Unicode, Unicode définit parfois plusieurs codes qui correspondent à des entités qui ne sont que des variantes visuelles (« glyphiques ») d'un même caractère. Unicode prévoit pour ces caractères des équivalences dites de compatibilité. L'apparence de ces caractères étant quelque peu différente, les remplacer par un autre caractère entraîne, en l'absence de balisage supplémentaire, une perte potentielle d'information de formatage. Le Mℓ disposé en carré – ㎖ (U+3396) – est une variante de compatibilité de M (U+006D) suivi de ℓ (U+2113). Unicode, rappelons-le (voir 4.9, Séquence équivalente), définit également des correspondances canoniques entre les caractères qui sont considérés comme identiques (et qui ne diffèrent donc même pas au niveau visuel). É (U+00C9) est une variante canonique de E (U+0045) + ◌́ (U+0301).

Unicode précise à partir de ces deux types d'équivalence deux formes de décomposition des caractères nécessaires quand on désire comparer des chaînes de caractères.

La *décomposition canonique* d'un caractère est réversible et n'entraîne aucune perte d'information. Elle peut donc être utilisée dans l'échange normalisé de textes. En effet, cette forme permet d'effectuer une comparaison binaire tout en conservant une équivalence canonique avec le texte non normalisé d'origine. La décomposition canonique d'une chaîne de caractères est la transposition successive et récursive de chaque caractère par sa valeur canonique correspondante jusqu'à ce que ces transpositions renvoient vers elles-mêmes, suivie de sa mise en ordre canonique.

La *décomposition de compatibilité* permet d'effectuer une comparaison binaire tout en conservant cette-fois-ci une équivalence de compatibilité avec le texte non normalisé d'origine. Cette dernière forme peut s'avérer utile en maintes occasions puisqu'elle permet d'éliminer des différences qui ne sont pas toujours pertinentes. Les caractères *katakana* à pleine chasse et à demi-chasse ont les mêmes décompositions de compatibilité et sont donc compatibles ; ils ne sont toutefois pas canoniquement équivalents puisque leurs chasses diffèrent. La décomposition de compatibilité d'une chaîne de caractères est la transposition successive et récursive de chaque caractère par sa valeur canonique et de compatibilité correspondante jusqu'à ce que ces transpositions renvoient vers elles-mêmes, suivie de sa mise en ordre canonique.

Ces deux formes transposent les caractères vers des caractères décomposés. Il existe également des formes de normalisation qui transposent les caractères vers des caractères composés (s'ils existent). Pour ce faire, il suffit de faire suivre les deux formes de décompositions introduites ci-dessus d'une composition canonique (E + ◌́ est donc transposé vers É).

En faisant suivre ou non les deux formes de décomposition par une composition canonique, on obtient un total de quatre formes de normalisation. Ces quatre formes

portent un nom (voir figure 16). Les deux formes normalisées vers les caractères précomposés se nomment C et KC. Le résultat de l'une est l'équivalent canonique du texte d'origine, le résultat de l'autre est son équivalent de compatibilité. Le K de KD et KC représente le mot *compatibilité* (suggéré par l'allemand *kompatibel*) alors que le C renvoie aux deux C de *composition canonique*. Le D lui se réfère à la décomposition.

	Sans composition canonique	Suivie d'une composition canonique
Décomposition canonique	D	C
Décomposition de compatibilité	KD	KC

Figure 16. *Les quatre formes de normalisation*

Les programmes doivent toujours considérer comme égales des chaînes Unicode qui sont canoniquement équivalentes. Une des façons les plus simples de s'en assurer est de normaliser toutes les chaînes de caractères, car si les chaînes sont normalisées leurs formes canoniques (C ou D) ont alors exactement la même représentation binaire. La forme de normalisation à utiliser dépend de l'application et de la plate-forme en question. La forme la plus fréquente est la forme C car cette forme canonique recomposée est compacte et d'ordinaire compatible avec les jeux de caractères préexistants, c'est d'ailleurs la forme choisie par le W3C pour les standards du Web. Les autres formes s'avèrent utiles dans d'autres contextes de comparaison textuelle. La figure 17 illustre les formes normalisées de la chaîne « affligé » dont on a codé le ffl sous la forme d'une ligature assez commune[26].

Nom	**Chaîne normalisée**
D	a + ffl + i+ g + e + ◌́
C	a + ffl + i+ g + é
KD	a + f +f + l + i+ g + e + ◌́
KC	a + f +f + l + i+ g + é

Figure 17. *Formes normalisées d'« affligé »*

26. Cette ligature est un caractère de compatibilité inclus à des fins de transcodage, Unicode recommande de l'éviter. Cette ligature peut toutefois être affichée (v. figure 22).

8. Ordonnancement et tri lexicographique

Trier des chaînes de caractères est un processus qui dépend fortement de la langue du lecteur-cible et même de l'application considérée. En effet, si l'allemand trie le « Ä » au début de l'alphabet, le suédois le trie après Z. Au sein d'une même langue, on n'ordonne pas toujours les chaînes de la même manière (il suffit de penser à l'ordre des chiffres et des noms propres dont les formes voisines sont regroupées dans un annuaire téléphonique). Toute norme de tri se doit de prendre en compte cette grande variabilité.

Le consortium Unicode (dans son rapport technique n° 10) et l'ISO (dans sa norme 14651[27]) ont défini des algorithmes de tri opérant sur des chaînes Unicode. Le rapport technique n° 10 d'Unicode (RTU-10) est un « profil » de la norme ISO/CEI 14651. Cette dernière a comme caractéristique principale de ne pas imposer de table de tri implicite. Pour qu'un processus soit conforme il faut explicitement déclarer un sous-tableau (un « delta ») pour la langue et pour la culture cibles. En cela, ISO/CEI 14651 reconnaît la nécessité d'adapter l'algorithme implicite à la culture et à la langue locales malgré une table qui propose au départ un modèle valable pour la majorité des langues. Dans cette optique, aucun tri n'est valable pour toutes les langues.

Le RTU-10 propose une table implicite à utiliser en l'absence d'autres indications et il n'insiste pas sur la notion d'adaptabilité, bien que certaines mises en œuvre intelligentes s'y astreignent sans doute par le jeu de la concurrence et de l'expression des besoins des acheteurs perspicaces. S'il existe une multitude de différences entre le RTU-10 et l'ISO/CEI 14561 en ce qui a trait aux caractères spéciaux, le résultat trié est essentiellement le même que celui obtenu avec la table de base de la 14651 en déclarant toutefois un delta minimal.

9. Rendu

9.1. Deux espaces distincts

Une police et le processus de rendu qui lui est associé définissent une correspondance arbitraire entre des valeurs Unicode et des glyphes. Chaque glyphe dans une police est identifié par un numéro de glyphe unique. Une table de transposition, appelée habituellement CMAP, permet d'effectuer le passage des numéros de caractères aux numéros de glyphes (voir la figure 18).

Une police se conforme à Unicode si sa CMAP transpose tous ou *une partie* des caractères Unicode vers des glyphes définis dans cette police. Il faut remarquer que tous les glyphes d'une police ne sont pas habituellement directement associés à un caractère Unicode. En effet, les variantes d'œil, les ligatures et les glyphes composés

27. ISO/CEI 14651, Classement international de chaînes de caractères — Méthode de comparaison de chaînes de caractères et description du modèle commun et adaptable de classement.

dynamiquement n'ont pas d'ordinaire d'entrée dans la table CMAP. Ces glyphes sont accessibles par l'application de transpositions définies dans d'autres tables définies dans les polices (par exemple GSUB pour OpenType et MORT pour AAT[28]). Remarquons que seule la CMAP connaît la valeur des caractères ; les autres tableaux d'une police sont tous indexés en fonction des numéros de glyphe et non des numéros de caractère. Une même police peut posséder plusieurs tables CMAP, chacune indexée selon un jeu de caractères différent.

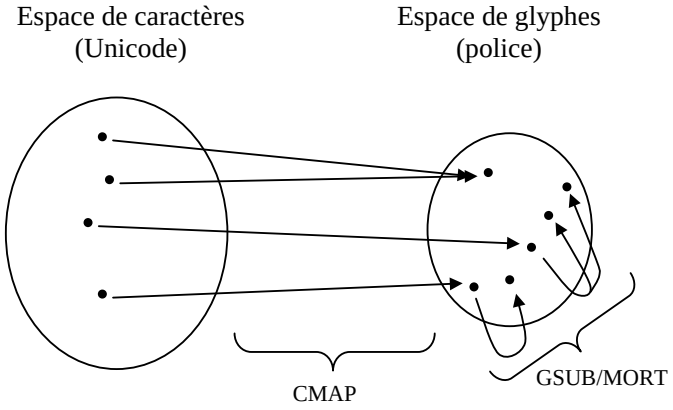


Figure 18. *Passage de l'espace de caractères à l'espace de glyphes*

La mise en correspondance entre les caractères stockés en mémoire et les glyphes ne constitue qu'une partie du rendu de texte. L'aspect final du texte rendu peut également dépendre du contexte (les caractères contigus dans la représentation en mémoire), des variations dans le dessin des polices utilisées ainsi que de paramètres de formatage (force du corps, exposant, indice et ainsi de suite).

Il existe pour toutes les écritures une relation archétypique entre les suites de caractères et les glyphes qui en résultent. Unicode explique en partie ces relations pour les écritures du monde, toutefois Unicode ne normalise pas cet aspect. En effet, une typographie fine peut nécessiter des règles bien plus détaillées que celles précisées par Unicode. Le standard Unicode documente la relation implicite qui existe entre les suites de caractères et leur dessin dans la seule intention qu'un même texte soit toujours stocké à l'aide d'une suite de codes de caractère identique et donc échangeable.

9.2. Processus de rendu et polices

Les polices de caractères comme OpenType comprennent un certain nombre de tables énumérées à la figure 19. OpenType est le nouveau format de police élaboré

28. Apple Advanced Typography, le nouveau nom de Quickdraw GX.

par Microsoft et Adobe. Il s'agit de la fusion des formats de police TrueType et PostScript, formats qu'OpenType est appelé à remplacer à terme. Des polices OpenType sont désormais livrées avec MS Windows. Le *Open* dans OpenType signifie que les fonderies peuvent ajouter leurs propres fonctions dans ces polices. On remarquera que les données définissant le contour des glyphes peuvent être incluses dans le format TrueType (table GLYF) ou PostScript (table CCF).

Table	Description
CMAP	Correspondances entre caractère et glyphe
GLYF	Dessin TrueType des glyphes (sous forme vectorisée)
CFF	Programme de police PostScript (définit l'œil des glyphes)
GSUB	Substitution de glyphes : 1-1, 1-*n* ou *n*-1
GPOS	Déplacement des glyphes et positionnement des glyphes les uns par rapport aux autres sur les deux axes (X,Y).

Figure 19. *Tables OpenType*

Parmi les tables ajoutées à TrueType pour former OpenType (GSUB, GPOS, GDEF), GSUB est sans doute la plus intéressante dans le cadre d'une discussion portant sur le modèle caractère-glyphe. Elle contient des informations que les applications-clientes (par exemple un traitement de texte) peuvent utiliser pour remplacer certains œils par d'autres. La table GSUB précise les œils en entrée et en sortie pour chaque substitution (voir figure 20), elle indique également le mode d'utilisation et l'ordre des substitutions. On peut définir un nombre quelconque de remplacements pour chaque écriture ou langue représentée dans la police. Ce dernier aspect permet ainsi d'éviter qu'une ligature « fi » ne se forme dans les textes turcs (langue qui distingue un i sans point du i pointé) tout en formant cette ligature pour les autres langues.

Type de remplacement	Exemple
Simple (forme verticale)	~ ➡ ~
Multiple (décomposition)	ﬄ ➡ f f l
Multiple (ligature)	ﻝ ﺍ ➡ ﻻ [29]
Variantes d'œil	& ➡ & & & &

Figure 20. *Exemples de substitution OpenType*

Précisons encore que la table GPOS permet de décaler les œils par rapport à leur position normale et de les placer les uns par rapport aux autres. Ceci est

29. En ordre logique.

particulièrement utile pour des écritures complexes dans lesquelles la position des éléments graphiques varie selon le contexte. C'est le cas en dévanâgarî où la table GPOS peut être utilisée pour ajuster la position précise des matras (des voyelles) et des autres signes diacritiques par rapport aux consonnes de base ou aux conjointes (les ligatures consonantiques). Les conjointes auront, pour leur part, été formées à l'aide de la table GSUB.

La figure 21 illustre un modèle de rendu textuel typique. Dans cet exemple, le texte-source Unicode est en forme normalisée KD, le résultat affiché pour sa part utilise la ligature « ffl » et un é précomposé. Le processus de rendu commence d'abord par exécuter l'algorithme bidirectionnel d'Unicode[30] dans l'espace des caractères. Il effectue ensuite le passage des caractères aux œils grâce à la CMAP. Pour chaque passage directionnel[31], il remplace les glyphes selon les règles définies par la table GSUB et, le cas échéant, des paramètres passés par l'application[32]. Ces remplacements se font de manière itérative : une ligature issue d'une substitution peut participer à une substitution ultérieure. Ensuite, une fois l'ordre et l'identité des œils déterminés, on calcule leur position relative précise à l'aide de la table GPOS. Enfin, le système d'exploitation utilise les tables GLYF ou CFF pour dessiner les œils.

La prise en charge des fonctionnalités originales d'OpenType, comme les substitutions, nécessite une modification des applications. C'est ce qui explique qu'il ne suffit pas de créer une police OpenType pour que MS Word affiche les ligatures latines qu'on aura pu ajouter à ces polices. En fait, aucun progiciel de Microsoft ne prend actuellement en charge les ligatures latines ! Microsoft n'utilise ces fonction–nalités GSUB que dans les langues non latines où elles s'avèrent essentielles. Seul Adobe, avec ses outils comme InDesign, semble prendre en charge les fonctions typographiques OpenType pour les écritures latines.

30. Unicode ne prescrit pas la mise en œuvre de cet algorithme, mais bien le résultat que la mise en œuvre doit produire.
31. C'est-à-dire dans une suite de caractères de même directionalité (droite-à-gauche, gauche-à-droite).
32. Dans le cas d'OpenType, les substitutions sont réalisées par le client et non le système d'exploitation. Le client a donc toute latitude pour sélectionner les substitutions désirées. En revanche, dans le cas du système ATSUI sur MacIntosh, tout le processus est effectué de manière transparente (ou opaque, c'est selon) par le système d'exploitation.

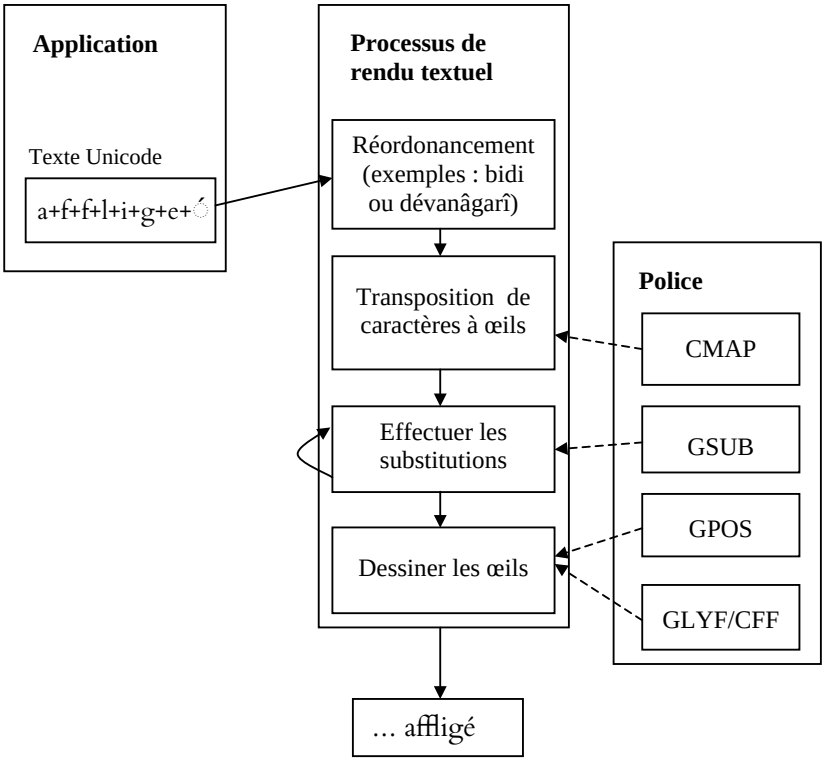


Figure 21. *Étapes internes du processus de rendu*

La figure ci-après illustre comment ce processus peut en pratique s'appliquer à une écriture complexe comme la dévanâgarî[33]. La conjointe djña (résultat de la suite ①②③) résulte d'une substitution de type ligature. Le placement du ④ juste en dessous de la hampe de la conjointe s'effectue à l'aide de la table GPOS. De même, le placement du ⑤ en dessous de (①②③ + ④) est une nouvelle fois réalisé à l'aide de la table GPOS. Sans celle-ci, cette barre souscrite aurait traversé le milieu du *diacritique ou* (④).

33. Pour plus de détails sur les règles de composition de la dévanâgarî, veuillez vous reporter à <http://iquebec.ifrance.com/hapax/pdf/Chapitre-10.pdf>

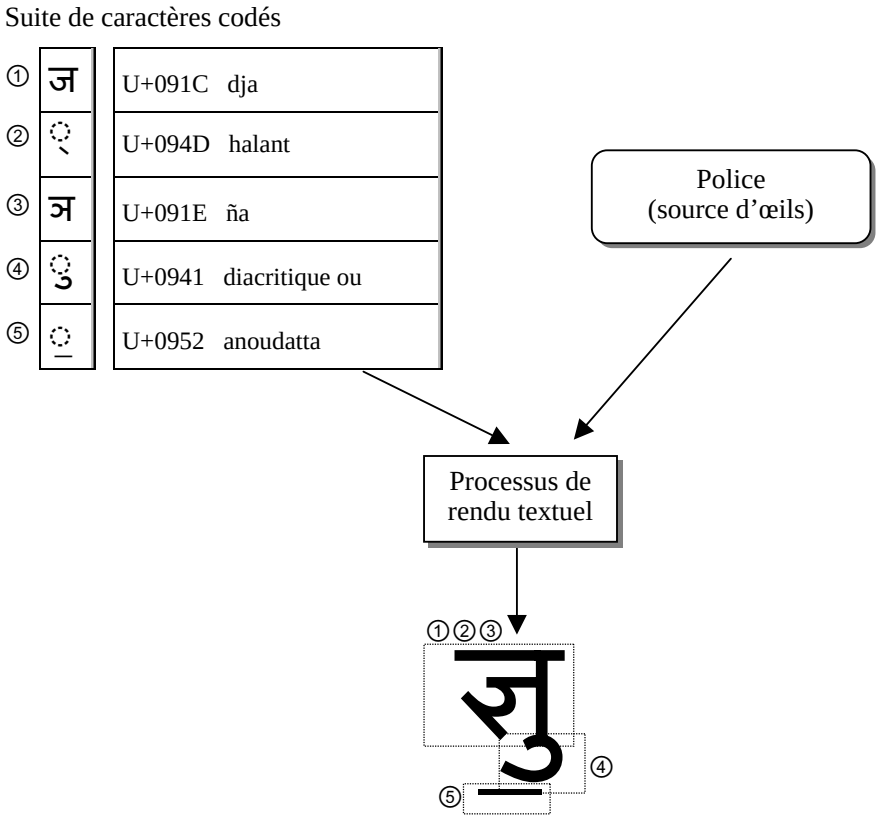


Figure 22. *Processus de rendu complexe*

10. Bibliographie

Le consortium Unicode, *The Unicode Standard, Version 3.0*, Reading, Addison-Wesley, 2000.

Le consortium Unicode, *The Unicode Standard, Version 3.1*, Mountain View, 2001, <http://www.unicode.org/Unicode/reports/tr27>.

Le consortium Unicode, *The Unicode Standard, Version 3.2*, Mountain View, 2002, <http://www.unicode.org/Unicode/reports/tr28>.

Le consortium Unicode, *Unicode Collation Algorithm*, Mountain View, 2002, <http://www.unicode.org/Unicode/reports/tr10>.

Le consortium Unicode, *Character Encoding Model*, Mountain View, 2000, <http://www.unicode.org/Unicode/reports/tr17>.

Patrick Andries, *Traduction et annotation du standard Unicode 3.0*, Longueuil, 2002, <http://hapax.iquebec.com>.

ISO/CEI 10646-1:2000, Jeu universel de caractères codés sur plusieurs octets (JUC) – *Partie 1: Architecture et plan multilingue de base*, Genève, 2000.

ISO/CEI 10646-1:2000/Amendement 1:2002, *Symboles mathématiques et autres caractères*, Genève, 2002.

ISO/CEI 10646-2:2001, Jeu universel de caractères codés à plusieurs octets (JUC) – *Partie 2: Plans complémentaires*, Genève, 2002.

ISO/CEI 14651:2001, Classement international et comparaison de chaînes de caractères – *Méthode de comparaison de chaînes de caractères et description du modèle commun d'ordre de classement*, Genève, 2002.