

RFC 3629 : UTF-8, a transformation format of ISO 10646

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 5 février 2007

Date de publication du RFC : Novembre 2003

<https://www.bortzmeyer.org/3629.html>

Autrefois, à l'époque du RFC 20¹, tout était simple, tous les textes étaient en anglais et n'avaient donc besoin que des quelques caractères d'ASCII. Mais le monde a changé, les étrangers ont voulu utiliser leurs langues bizarres et Unicode est apparu. Seul problème, les encodages d'Unicode n'étaient pas tout à fait adaptés aux spécificités du monde Internet, et voilà pourquoi un RFC a finalement été écrit pour apporter au monde... UTF-8.

Unicode permet de représenter toutes les écritures du monde. Une des particularités de ce jeu de caractères est qu'il sépare le jeu lui-même, la liste des caractères et leurs noms, de l'encodage en bits de ces caractères (voir mon exposé à JRES <<http://2003.jres.org/diapo/paper.3.pdf>>). Un même texte en Unicode peut être encodé en UTF-16, UTF-32, Punycode et bien d'autres encore. Chacun de ces encodages a des avantages et des inconvénients. La démarche qui a mené à UTF-8 venait de l'importance donnée à certains critères :

- Compatibilité avec l'ASCII, et notamment avec toutes les bibliothèques existantes qui s'attendent à recevoir de l'ASCII (comme les fonctions de la libc sur Unix),
- Minimisation de l'occupation mémoire dans les cas des textes écrits avec l'alphabet latin.

UTF-8 est un encodage de taille variable : un caractère Unicode est représenté avec un nombre d'octets compris entre un et six. Sa principale particularité est qu'un caractère du jeu ASCII est représenté en UTF-8 comme en ASCII. Tout fichier ASCII est donc un fichier UTF-8 (l'inverse n'étant évidemment pas vrai). De même, tout caractère dont le bit de plus fort poids est à zéro est forcément un caractère ASCII, ce qui permet d'utiliser la libc (qui s'attend à trouver des chaînes de caractères terminées par le caractère NUL).

Le support d'UTF-8 dans les logiciels aujourd'hui est variable : excellent dans les navigateurs Web, il est assez bon dans les SGBD (voir par exemple PostgreSQL <<https://www.bortzmeyer.org/postgresql-unicode.html>>), moyen dans les éditeurs et parfois absents de certains outils de base

1. Pour voir le RFC de numéro NNN, <https://www.ietf.org/rfc/rfcNNN.txt>, par exemple <https://www.ietf.org/rfc/rfc20.txt>

(comme le générateur d'analyseurs lexicaux Lex). C'est ainsi que certains utilisateurs ont du mal à passer à UTF-8 <<https://www.bortzmeyer.org/pas-encore-utf8.html>>.

Dans certains langages de programmation, comme Python ou Perl, manipuler de l'Unicode (et, entre autre, lire et écrire de l'UTF-8) est aujourd'hui très simple alors que d'autres comme Haskell, Ruby, C ou PHP n'y sont pas encore vraiment.

Normalisé à l'origine dans le RFC 2044, UTF-8 est devenu le « nouvel ASCII ». Suivant le RFC 2277, presque tous les protocoles Internet qui manipulent de l'Unicode y font référence (la principale exception étant IDN, qui préfère l'encodage Punycode décrit dans le RFC 3492).