Polytech-Lille
Le codage UTF-8

François Boulier

25 octobre 2011

1 Travail à réaliser

Écrire un programme C qui vérifie qu'un fichier texte vérifie bien le format UTF-8. Le nom du fichier à tester doit être récupéré sur la ligne de commande. Il peut éventuellement être précédé de l'option « `-v` » pour provoquer un affichage « verbeux ».

Si l'option « `-v` » n'est pas précisée, le programme doit recopier le contenu du fichier sur la sortie standard et imprimer un « `replacement character` » (codage UTF-8 `0xef 0xbf 0xbd`, valeur `0xfffd`) à la place de tout caractère non UTF-8. Le programme doit retourner 0 si aucune erreur n'a été détectée, 1 sinon.

Si l'option « `-v` » est précisée, le programme doit recopier le contenu du fichier sur la sortie standard et imprimer davantage d'informations en cas d'erreur (le codage hexadécimal de la séquence d'octets fautive et une raison pour laquelle la séquence est considérée comme fautive). De plus, en fin d'analyse, le programme doit imprimer le nombre d'erreurs rencontrées ainsi qu'un caractère UTF-8 rencontré parmi ceux qui comportent le plus d'octets.

Voici quelques exemples.

```
# Message à afficher si le nombre d'arguments est incorrect
boulier@ciney:~/utf8$ ./a.out
usage: ./a.out [-v] file-name

# Le programme à réaliser
boulier@ciney:~/utf8$ ./a.out -v utf8-tester.c
[...]
longest encoding: 3 bytes [€]  e2 82 ac
number of errors: 0

# Un fichier plein d'erreurs
./a.out -v UTF-8-test.txt
[...]
5.3.2 U+FFFF = ef bf bf = "[noncharacter: ef bf bf]"                    |
```

```
                                                                              |
THE END                                                                       |
longest encoding: 4 bytes [ ]  f0 90 80 80
number of errors: 216

# Le même sans l'option -v (le « replacement character » apparaît)
5.3.2  U+FFFF = ef bf bf = " "                                                |
                                                                              |
THE END                                                                       |
boulier@ciney:~/utf8$ echo $?
1
```

2 Consignes

Lors de la lecture du fichier, on demande que chaque caractère lu soit stocké dans une variable du type suivant. On demande aussi d'implanter les fonctions dont les prototypes suivent.

```
/*
 * Structure pour stocker un caractère UTF-8
 */

#define NBMAX_BYTES 8
struct character
{   unsigned char bytes [NBMAX_BYTES]; /* les octets */
    int nbbytes;                       /* le nombre d'octets */
};

/* initialisation à la séquence vide */
void init_character (struct character* c);

/* ajout d'un nouvel octet en fin de séquence */
void append_byte (struct character* c, unsigned char byte);

/* imprime la séquence d'octets, c'est-à-dire, le caractère */
void print_character (struct character* c);

/* imprime la séquence d'octets en hexadécimal */
void dump_character (struct character* c);
```

3 Le codage UTF-8

Un caractère UTF-8 est codé sur 1 à 4 octets. Il a une *valeur* comprise entre `0x00` et `0x10ffff`.

Un caractère codé sur 1 octet est un code ASCII. Sa valeur est donc comprise entre `0x00` et `0x7f`.

Si un caractère est codé sur 2, 3 ou 4 octets, alors chaque octet est composé d'une suite de bits de contrôle, puis de bits destinés à former la valeur du caractère.

Soit un caractère codé par n octets ($2 \leq n \leq 4$). Le premier octet commence par n bits à 1, suivi d'un bit à 0, suivi de $7 - n$ bits de valeur. Les autres octets commencent par les deux bits 10, suivi de 6 bits de valeur. La valeur du caractère s'obtient en concaténant les bits de valeur. Exemple :

- 2 octets. Codage : `110xxxxx 10yyyyyy`. Valeur : `xxx xxyyyyyy`.
- 3 octets. Codage : `1110xxxx 10yyyyyy 10zzzzzz`. Valeur : `xxxxyyyy yyzzzzzz`.
- 4 octets. Codage : `11110xxx 10yyyyyy 10zzzzzz 10wwwwww`. Valeur : `xxxyy yyyyzzzz zzwwwwww`.

Certains codages sont interdits :

- le codage d'une valeur doit se faire sur un nombre minimal d'octets :
  1. les premiers octets `0xc0` et `0xc1` sont interdits (ils donneraient lieu à un code ASCII sur deux octets) ;
  2. si le premier octet est `0xe0`, la valeur finale doit être supérieure ou égale à `0x800` ;
  3. si le premier octet est `0xf0`, la valeur finale doit être supérieure ou égale à `0x10000`.
- la valeur ne doit pas dépasser `0x10ffff` ;
- les valeurs comprises entre `0xd800` et `0xdfff` sont réservées pour coder de l'UTF-16 et ne correspondent à aucun caractère UTF-8 ;
- il existe 66 valeurs pour des « non-caractères » : les valeurs comprises entre `0xfdd0` et `0xfdef` ainsi que les 34 valeurs terminant par `0xfffe` et `0xffff` (de `0xfffe`, `0xffff` à `0x10fffe`, `0x10fff`).