R1.03 - Architecture des ordinateurs
TP1 - Codage des caractères

Partie 1 :

4. Comparez les représentations hexadécimales des fichiers. Est-ce-que vous constatez une différence entre les deux codages ? Y a-t-il des caractères dont le codage diffère de l'ASCII ?

Il n'y a pas de différence de représentations décimales entre le fichier encoder en UTF-8 et l'ISO 8559-15.
Tout les caractères que contient ce document font partie du codage ASCII.

5. Utilisez « du » pour afficher la taille en octets (bytes) de chaque fichier. Y a-t-il une différence de taille entre les deux fichiers ? Justifiez.

Après avoir utilisé la commande « **du -d utf8.txt** » et la commande « **du -d iso.txt** », on constate qu'il n'y a aucune différence entre la taille en octets des deux fichiers. L'encodage UTF-8 et l'ISO 8559-15 font 16 octets chacun.

Partie 2 :

4. Comparez les représentations hexadécimales des fichiers. Est-ce qu'il y a des caractères ayant des codages différents ? Si oui, quels sont ces caractères ? Indiquez leurs codes hexadécimaux pour chaque codages. Y a-t-il des caractères dont le codage diffère de l'ASCII ? Pourquoi certains octets ne sont pas affichable en ASCII ?

Les caractères ayant un codage different de l'ASCII sont : le « é » et le « € ».
La représentation hexadécimal de « é » en UTF-8 est : **c3 a9**
La représentation hexadécimal de « é » en ISO est : **e9**
La représentation hexadécimal de « € » en UTF-8 est : **e2 82 ac**
La représentation hexadécimal de « € » en ISO est : **a4**

Dans le codage ASCII il y a uniquement les caractères de base. Tout les autres caractères sont ajoutées avec d'autres type d'encodage.

5. Utilisez du pour afficher la taille en octets (bytes) de chaque fichier. Y a-t-il une différence de taille entre les deux fichiers ? Si oui expliquez la différence ?

Le fichier ISO_Special fait 19 Octets pour 19 caractères.
Le fichier UTF8_Special fait 22 Octets pour 19 caractères.

Il y a une différence de taille de fichier car l'encodage ISO 8519-15 utilise 1octet/caractère contrairement à l'UTF8 qui utilise 1octet par caractère pour les caractères ASCII mais utilise 1,5 octets pour les caractères non disponible dans l'encodage ASCII.

6. Dans Gedit, remarquez comment le texte est interprété différemment en utilisant les codages UTF-8 et ISO-8859-15 (latin-9) et relevez les conséquences d'une interprétation erronée.

L'encodage du fichier ISO en UTF-8 :

Les caractères « é » et « € » sont transformés respectivement en « \E9 » et « \A4 ».

L'encodage du fichier UTF-8 en ISO :

Les caractères « é » et « € » sont transformés respectivement en « Ã© » et « à (Un carré avec des écritures à l'intérieur qu'on arrive pas à refaire) ».

Partie 3 :

4. Observez les représentations hexadécimales des caractères dans le fichier UTF-16. Comparez les représentations hexadécimales des caractères du fichier UTF-16 avec celles du fichier UTF-8. Qu'est ce que vous constatez ?

Il y a les octet "fffe" au début du fichier UTF-16
Les caractère sont composée sur 2 octet '4a' pour J en UTF-8 alors que sur UTF-16 les caractère sont sur 4 octet '4a00 ' pour J

```
00000000: fffe 4a00 6500 2000 7400 2700 6100 6900  ..J.e. .t.'.a.i.
00000010: 2000 6400 6f00 6e00 6e00 e900 2000 3100   .d.o.n.n... .1.
00000020: 2000 ac20 2100 0a00
```

5. Quels sont les caractères codés différemment entre UTF-16 **et** UTF-8 **?**

Le ‘é’ qui est en utf-8 codé comme ça ‘c3a9’ et en utf-16 ‘e900’

Le ‘€’ qui est en utf-8 codé comme ça ‘e2 82ac’ et en utf-16 ‘ac20’

6. Indiquez si le codage est en big-endian **ou en** little-endian**. Donnez les codes des caractères** é **et** € **en décimal.**

L’UTF-8 et l’UTF-16 sont différentes car en UTF-16 notre ordinateur affiche l’encodage en Little Endian, qui donc écrit d’abord l’octets de poids fort avant l’octet de poids faible.

7. (facultatif) **Vous verrez probablement les octets** "fffe" **au début du fichier** UTF- 16 **lorsque vous utilisez la commande** xxd**. Expliquez à quoi sert le** "fffe" **au début du fichier.**

D’après Wikipédia et des constatations que nous avons effectué le « fffe » indique dans quelle sens sont codées l’informations. Donc si on a un « fffe » ça veut dire qu’on est sur une machine Little-Endian (l’exemple des puces Intel). Et si on a un « feff » la machine traite les informations en Big-Endian donc l’octet de poids faible viendra avant l’octet de poids forts.