11010010110101010001010111010011010110 1sin$^2\,x+\cos^2\,x=1$0010101011010010101101000011110001010000111001e$^{i\pi}$+1 = 0101011010010101011010010101101010101011010110 10**Tilleuil**0101011010101010111111110110100101011010100101101101010101101 1$\sum_{n=1}^{\infty}\frac{1}{n^2}=\frac{\pi^2}{6}$1010110100101011010110100101010110100101010110 0100101100101 0**Desbonnez**1101010101001101101010 1$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$0101101011011010100101011010011010110100101101101101 101 **InforMathématique**0101000101101101001011011010010 1**informathematiqueptijmd@gmail.com**1010010101101010101010010101101 0**www.informathematique.be**

UTF-8, MacOsRoman, ISO-Latin1 : conversions d'encodage de fichiers de type texte

Le français est une très belle langue, mais pose parfois des problèmes au monde informatique.

Nous présentons ici des outils de conversions pour des fichiers de type texte codés en ISO-Latin1 et MacOsRoman vers du codage UTF-8, plus universel.

Un fichier « de type texte » est un fichier qui ne contient… que du texte ; aucune mise en forme, ni mise en page ; il est créé à l'aide d'un « éditeur de texte ». On ne parle donc pas ici de « traitement de textes » comme *Microsoft Word*, *OpenOffice Writer*, ou autre.

Les éditeurs de texte servent essentiellement à faire du codage de langages informatiques, et chacun d'eux a bien souvent son éditeur associé qui s'occupe de la coloration syntaxique, voire de l'exécution des instructions.
Les noms de ces fichiers ont des extensions qui, bien souvent, sont associées au langage : *.py* pour Python, *.rb* pour Ruby, *.tex* pour LaTeX, *.html* (1) pour Internet, pour n'en citer que quelques uns, sans oublier *.txt*, extension par défaut proposée par des éditeurs standards comme *TextEdit.app* chez Mac et *Notepad.exe* chez Windows.

1. Introduction

En informatique, il existe pléthore de systèmes d'encodage des caractères, le plus connu est certainement ASCII (2), datant du début des années 60, et mis au point pour la langue anglaise, et donc pas de caractères accentués ni de caractères spécifiques à une autre langue.

Pour faire (très) bref et court-circuiter toutes les évolutions, cohabitent dans nos systèmes actuels en Europe occidentale, trois types d'encodages courants :

- MacOs Roman : uniquement sous MacOsX ;
- ISO-Latin1 (iso-8859-1) : multi plateforme (3), surtout sous Windows ;
- UTF-8 : l'universel (créé en 1991).

UTF-8 permet de mélanger plusieurs langues dans un même document, et donc de régler tous les problèmes de comptabilité entre différents systèmes.

Vous ne le savez peut-être pas, mais vos *éditeurs de textes* sont configurés pour l'un de ces trois types d'encodage. C'est transparent pour l'utilisateur, et sans importance s'il n'y a pas d'échanges de documents de type texte avec d'autres utilisateurs, à moins bien sûr qu'ils utilisent le même codage que vous.

(1). HyperText Markup Langage.
(2). American Standard Code for Information Interchange.
(3). MacOsX, Windows, Linux.

Le type de codage par défaut de vos éditeurs de textes est visible dans les options ou préférences de configuration. À consulter *absolument* !

> ISO-Latin1 et MacOs Roman sont incompatibles avec UTF-8 ! Tous les caractères accentués disparaissent, ou sont remplacés par des « hiéroglyphes », si un éditeur configuré en UTF-8 ouvre un document codé en ISO-Latin1 ou en MacOs Roman, et *vice-versa.*

Il est donc très important de connaître le type d'encodage de votre éditeur de textes, ainsi que celui de la personne avec qui vous échangez.

Le principe est simple. Vous habitez à Java, vous écrivez des textes en javanais et vous les échangez avec d'autres javanais : tout va bien !
Vous envoyez un texte en javanais à un ami chinois : il ne comprend rien !

Vous avez compris ? Il faut une traduction, bien sûr.

2. Un outil de conversion

Sous UNIX, LINUX et MacOsX [4], la commande `iconv` permet de convertir un fichier texte d'un type d'encodage vers l'autre.

C'est une commande UNIX, et il faut donc utiliser le *Terminal* pour l'exécuter. La commande va créer un second fichier dans l'encodage spécifié. Conseil d'ami : faites toujours une copie de l'original avant toute manipulation.

Syntaxe :

- de ISO-Latin1 vers UTF-8 :
  `iconv -f latin1 -t utf8 fichier_latin1.txt > fichier_utf8.txt`
- de UTF-8 vers ISO-Latin1 :
  `iconv -f utf8 -t latin1 fichier_utf8.txt > fichier_latin1.txt`
- de MacOs Roman vers ISO-Latin1 :
  `iconv -f mac -t latin1 fichier_MacOsRoman.txt > fichier_latin1.txt`
- de MacOs Roman vers UTF-8 :
  `iconv -f mac -t utf8 fichier_MacOsRoman.txt > fichier_latin1.txt`

Dans la syntaxe qui précède, on a choisi l'extension standard des fichiers de type texte (*.txt*), mais dans la pratique, peu importe l'extension ; dans la suite on parlera plus particulièrement des fichiers LaTeX, d'extension *.tex*.

Lorsqu'on tape l'instruction sur la ligne de commande du terminal, il faut que le dossier contenant le fichier à traduire soit le **dossier actif**. Or, lorsqu'on travaille sur un document, il est bien souvent dans un dossier profond dans la hiérarchie.

Le plus simple est de créer *sur le bureau* un dossier appelé par exemple *conversion* [5] dans lequel on copie le fichier à convertir.

Ainsi, dans le *Terminal*, on tape d'abord `cd desktop/conversion`

puis `iconv ...`

(4). Pour nos amis Windoziens, voir plus loin.
(5). Ou Jules, Marcel, scoubidou, ...

3. Le cas LaTeX

Dans le préambule d'un fichier source .tex, il y a la commande qui stipule dans quel type d'encodage travaille votre éditeur : `\usepackage[latin1]{inputenc}` s'il est configuré en ISO-Latin1, `\usepackage[utf8]{inputenc}` s'il est configuré en UTF-8 et `\usepackage[applemac]{inputenc}` s'il est configuré en MacOs Roman.

Cette commande va souvent de pair avec `\usepackage[T1]{fontenc}`.

> Lorsque vous ouvrez un fichier *.tex* issu d'un échange, et dont vous avez converti le codage afin de le rendre compatible avec votre éditeur, n'oubliez pas de modifier l'option de `inputenc` AVANT de le compiler, sous peine de gros problèmes, allant jusqu'à l'impossibilité de compiler le texte !

Le cas XeLaTeX

Si vous êtes amené à faire une compilation en XeLaTeX, par exemple si votre document contient du code Pstricks, les deux *package* `fontenc` et `inputenc` sont à remplacer par

`\usepackage{fontspec}`

Le codage UTF-8 est obligatoire pour le XeLaTeX !

4. Pour nos amis windoziens

Sur le net j'ai trouvé une instruction MSDOS pour la conversion ISO-Latin1 —> UTF 8.

Elle est à exécuter via la ligne de commande, accessible, si mes souvenirs sont exacts, par le fichier `command.exe`, l'équivalent Windows du *Terminal* MacOsX.

```
"Get-Content .\testLatin1.txt" > testUtf8.txt
```

Comme je ne suis pas (plus) utilisateur de Windows, je ne sais pas tester l'instruction.

Il est aussi possible d'exécuter du code UNIX sous Windows (depuis la version 10) ; j'ai aussi trouvé sur le net
`https://www.eugenetoons.fr/utiliser-le-shell-bash-de-linux-sous-windows-10/`

5. Connaître le type d'encodage d'un fichier sans l'ouvrir

Commande **super importante** qui permet d'éviter une ouverture malencontreuse !

Sur la ligne de commande du terminal, et en supposant que nomfichier soit dans le répertoire actif, taper la commande [6]

```
file -mime-encoding nom_du_fichier
```

Résultats fournis par la commande :

(6). Plus loin, un script qui permet de tester tous les fichiers d'un répertoire.

- si le texte est codé en MacOs Roman —> *unknown-8bit*
- si le texte est codé en ISO-Latin1 —> *iso-8859-1*
- si le texte est codé en UTF-8 —> *utf-8*

Tout ce qui est écrit précédemment convient très bien pour traiter quelques fichiers. Mais cela peut devenir très lourd si on décide de changer son fusil d'épaule et de travailler dorénavant en UTF-8, et qu'il y a des centaines de fichiers à traiter...

Il peut donc être intéressant de faire ce qu'on appelle un traitement par lot, ou encore un traitement de masse.

On développe ci-après la conversion de MacOs Roman vers UTF-8 ; celle de ISO-Latin1 vers UTF-8 s'en déduit très facilement.

6. Convertir tous les fichiers d'un répertoire

Il est nécessaire de passer par la programmation et de créer un script qui va, dans une boucle, automatiser la commande de conversion. L'exemple décrit ci-après permet des conversions de MacOsRoman vers UTF-8. On expliquera la manière de le modifier pour des conversions de Iso-Latin1 vers UTF-8.

Le plus simple est de continuer à travailler dans le dossier *conversion* que nous avons créé sur le bureau. On y place tous les fichiers à convertir, on supposera qu'ils ont tous l'extension *.tex*.

Dans le même dossier, on placera le fichier script, que l'on nommera RomanUtf8 [7], attention, sans extension ! Il est à encoder simplement avec un éditeur de texte. Voici le code :

```
#! /bin/bash
set -e

# RomanUtf8
# conversion des fichiers MacOsRoman.tex du répertoire actif
# en UTF-8 avec préfixe "utf8-"

for fichier in *.tex; do
    iconv -f mac -t utf8 $fichier > "utf8-$fichier";
done
```

Tous les caractères #, excepté le premier, introduisent une ligne de commentaire, c'est-à-dire du texte libre, non exécutable. Les lignes qui en sont précédées ne sont donc pas obligatoires, mais elles permettent d'expliquer ce qui se passe.

Pour exécuter le script, il faut

- démarrer le Terminal
- activer le répertoire *conversion* : `cd desktop/conversion`
- exécuter le script : `bash RomanUtf8`

Tous les fichiers convertis gardent leur nom d'origine, mais avec le préfixe *utf8-*. Ainsi, le fichier *interro17.tex*, codé en MacOsRoman, va donner un *utf8-interro17.tex*.

(7). Pour les noms de fichiers, UNIX n'est pas sensible à la casse ; l'utilisation de majuscule peut simplifier la lecture du nom.

Pour convertir des fichiers codés en Iso-Latin1, il suffit de remplacer dans le script le mot mac par latin1. Il est alors préférable de modifier le nom du script, comme par exemple Latin1Utf8.

Et on peut encore aller plus loin : modifier, dans les nouveaux fichiers maintenant codés en UTF-8, la commande `\usepackage[applemac]{inputenc}` en `\usepackage[utf8]{inputenc}`.

Le plus sage est de créer un second script, ce qui permettra d'utiliser soit l'un, soit l'autre, selon ce que l'on veut faire.

Voici donc RomanUtf8inputenc :

```
#! /bin/bash
set -e

# RomanUtf8inputenc
# conversion des fichiers MacOsRoman.tex du répertoire actif
# en UTF-8 avec préfixe "utf8-"
# et remplacement de \usepackage[applemac]{inputenc}
# par \usepackage[utf8]{inputenc}

for fichier in *.tex; do
    iconv -f mac -t utf8 $fichier > "utf8-$fichier";
done

for fichier in utf8-*.tex; do
    sed -i -e "s/applemac/utf8/g" "$fichier";
done

for fichier in utf8-*-e; do
    rm $fichier;
done
```

- La première boucle fait les conversions et ajoute le préfixe « utf8- » au nom du fichier ; c'est RomanUtf8.
- La deuxième remplace dans chaque nouveau fichier le mot *applemac* par *utf8*, ce qui modifie la commande `\usepackage[applemac]{inputenc}`.
  Si par hasard le document contient le mot *applemac* ailleurs, il sera aussi modifié, c'est le talon d'Achille… mais peu probable.
  Pour des conversions de Iso-Latin1, modifier mac et applemac en latin1 ; et tant qu'à faire, modifier également le nom du script en Lati1Utf8inputenc.
- La troisième efface des fichiers temporaires créés par la deuxième boucle.

Remarques

1. Les noms des répertoires et des scripts ont été choisis ici pour les explications ; chacun fait évidemment son lit comme il se couche !
2. Les scripts qui précèdent sont orientés LaTeX avec des fichiers d'extension *.tex*. Ils peuvent s'appliquer à des fichiers « standards » d'extension *.txt*, et n'importe quel autre. Il suffit de faire la petite modification de l'extension dans le script. La modification de la commande `\usepackage[...]{inputenc}` n'y a pas de sens dans ce cas, et donc la première boucle suffit.

Attention : les commandes de manipulations de fichiers (et donc les scripts qui précèdent), que ce soit en MSDOS (Windows) ou en UNIX (Linux et MacOsX) ont horreur des noms de fichiers contenant des espaces, des caractères accentués et des caractères spéciaux ! Utilisez uniquement des caractères alphanumériques, et éventuellement le trait de soulignement. C'était déjà la règle au tout début de l'informatique, et à l'époque c'était maximum 8 caractères !

7. Tester le type d'encodage de tous les fichiers du répertoire actif

Il peut être intéressant de connaître le type d'encodage de tous les fichiers d'un répertoire, sans cibler particulièrement les fichiers d'extension *.tex*.

Le script qui suit est appelé TypeEncodage, il suffit de le placer dans le répertoire actif.

```
#! /bin/bash
set -e

# TypeEncodage
# Affiche le type d'encodage de chaque fichier
# du répertoire actif

for fichier in *; do
    type_encod=$(file --mime-encoding $fichier);
    echo $type_encod;
done
```

Comme pour les scripts précédents,

- démarrer le Terminal
- activer le répertoire *rep* : `cd desktop/rep`
- exécuter le script : `bash TypeEncodage`

Outre les trois types qui nous ont intéressés précédemment, on peut trouver par exemple *us-ascii*, *binary*, etc.

8. Un utilitaire... utile : supprimer les espaces dans les noms de fichiers

Il ne contient qu'une seule instruction, mais pas simple à taper ; c'est donc plus facile de taper le nom du script dans la ligne de commande plutôt que de taper l'instruction.

L'instruction supprime tous les caractères « espace » dans les noms de fichiers du répertoire actif et les remplace par le caractère de soulignement. Voici NomSansEspace :

```
#! /bin/bash
set -e

# NomSansEspace
# supprime les espace dans les noms des fichiers
# du répertoire actif et les remplace par _

find . -type f -name "*" -exec bash -c 'mv "$0" "${0// /_}"' {} \;
```

À placer dans le répertoire à traiter, qui doit être actif, et à exécuter… comme les autres.