CC

Zeste de savoir

C++, Windows et l'UTF-8

15 janvier 2019

Table des matières

1. Introduction

Aujourd'hui, en pleine rédaction de la suite du tutoriel C++, et notamment du chapitre sur les fichiers, je me suis intéressé à l'encodage des chaînes de caractères et la manipulation d'UTF-8 en C++ moderne. Je précise que je suis sous Windows, parce que GNU/Linux a le bon goût de marcher correctement.

D'abord s'est posée la question de savoir comment lire et écrire des caractères français, russes ou autres dans un fichier. Ça, c'est simple, il suffit d'utiliser `std::string` et de préfixer les chaînes de `u8`. Exemple.

```cpp
#include <fstream>
#include <iostream>
#include <string>
#include <vector>

int main()
{
    std::vector<std::string> const phrases
    {
        u8"Voici un mot important.\n",
        u8"Russe : резиновые перчатки",
        u8"Polonais : gumowe rękawiczki",
        u8"Roumain : mănusi de cauciuc"
    };

    std::ofstream fichier { "sortie.txt" };
    for (auto const & phrase : phrases)
    {
        fichier << phrase << std::endl;
    }

    return 0;
}
```

Puis je me suis demandé comment afficher dans la console du texte non-ASCII. Je précise que ma police de caractère est **Lucida Console**, donc compatible. Essayons donc.

```cpp
#include <iostream>
#include <string>
#include <vector>

int main()
{
    std::vector<std::string> phrases
    {
        u8"Voici un mot important.\n",
        u8"Russe : резиновые перчатки",
        u8"Polonais : gumowe rękawiczki",
        u8"Roumain : mănusi de cauciuc",
        u8"Japonais : "
    };

    SetConsoleOutputCP(CP_UTF8);
    std::cout << phrases[1] << std::endl;

    return 0;
}
```

```
Russe : резиновые перчатки
```

Cool, c'est plutôt bon signe. Essayons le japonais avec une police **NSimSun**, parce que Lucida Console ne sait pas afficher le japonais.

```
Japonais :
```

Ah, donc l'affichage, on dirait bien que ça marche. Et si on essaye l'inverse ? Essayons d'entrer du texte UTF-8. Commençons simple, commençons par notre langue à nous.

```cpp
#include <iostream>
#include <string>
#include <Windows.h>

int main()
{
    SetConsoleOutputCP(CP_UTF8);
    SetConsoleCP(CP_UTF8);

```

```
    std::string text;
    std::cin >> text;
    std::cout << "Donc on a : " << text << std::endl;

    return 0;
}
```

```
é
Donc on a :
```

Arf, comment ça rien ? Non mais oh ! Et en cherchant sur StackOverflow, on tombe sur ça.

> Be aware that there are many bugs with UTF-8 as codepage. Most are WONTFIX.
>
> *Deduplicator*

Et en creusant un peu (désolé je ne retrouve plus le lien), certains disent que, même avec Windows 10, cette fonction se contente de planter silencieusement quand on lui passe le *code page* correspondant à l'UTF-8 et qu'on lui donne des caractères non-ASCII.

À noter que si j'utilise le *code page* 866 pour le cyrillique, il n'y a aucun problème à taper des lettres russes, tout comme 850 marche pour nos lettres françaises. Voici la liste des *codes pages*.

Tout ça pour dire que C++ et Windows, c'est parfois compliqué. Et sans verser dans le troll, bein Linux des fois c'est clairement supérieur.

PS : avant de se quitter, jetez un oeil au logo de ce billet et dites moi en toute honnêteté si vous préférer celui-là ou bien celui du tutoriel C++ (ci-dessous). J'ai simplement fait un logo au couleur de notre cher site.

Contenu masqué n°1

Contenu masqué

Contenu masqué n°1

FIGURE 1. – Logo actuel du tutoriel C++.

Contenu masqué

Retourner au texte.