Project: Programming Language - C++ (WG21)
Author: Andrew Tomazos <andrewtomazos@gmail.com>
Date: 2020-06-11

Analysis of C/C++ Source File Character Encoding

Abstract

2,683,676 unique C/C++ source files from the ACTCD19 dataset (https://codesearch.isocpp.org) were studied to determine their source character encoding and for the presence of a UTF-8 byte-order-mark (BOM).

Method

For the purpose of this study source files are considered identical if they consist of identical bit sequences. Of each set of such identical files all but one was discarded.

Each source file was then analyzed to determine:

- whether or not it starts with a BOM (0xEF 0xBB 0xBF).

- excluding any BOM, whether the the file has any high bit set (0x80), if not it was classified as ASCII

- if not classified as ASCII, whether the file is well-formed UTF-8. Each non-ASCII character in the file must be a valid UTF-8 starting code unit trailed by the correct number of UTF-8 trailing code units. ie the file is a concatenation of octet sequences that fit one of the masks given in RFC 3629:

```
>   Char. number range  |        UTF-8 octet sequence
```

```
>      (hexadecimal)      |              (binary)
>    --------------------+---------------------------------------------
>    0000 0000-0000 007F | 0xxxxxxx
>    0000 0080-0000 07FF | 110xxxxx 10xxxxxx
>    0000 0800-0000 FFFF | 1110xxxx 10xxxxxx 10xxxxxx
>    0001 0000-0010 FFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
```

- Otherwise, the file was classified as LATIN1

Results

The results of this analysis were as follows:

	BOM	NO BOM	TOTAL
ASCII	844	2542094	2542938
UTF8	227	122705	122932
LATIN1	0	17806	17806
TOTAL	1071	2682605	2683676

We would highlight the following observations:

- ASCII is 20x more common than UTF-8
- UTF-8 is 7x more common than LATIN1
- Only 1 in every 500 UTF-8 source files have a BOM
- 80% of the time a BOM is present, it is present on an otherwise ASCII file

Steps to reproduce

1. Download and extract the raw ACTCD19 source files from https://codesearch.isocpp.org/actcd19.tar.gz (10.5 GB)
2. Run fdupes program to deduplicate
3. Run the following C++ program:

Makefile

```
process: process.cc Makefile
	g++ -Wall -std=c++17 process.cc -lstdc++fs -O3 -o process
```

process.cc

```
#include <filesystem>
#include <fstream>
#include <iostream>
#include <streambuf>
#include <string>

#define log(var)                                                                \
  do {                                                                          \
    std::cout << #var << " = " << (var) << "\n";                                \
  } while (0)

std::string slurp(const std::filesystem::path &p) {
  std::ifstream f(p.string(), std::ios::in | std::ios::binary);
  std::string s;
  f.seekg(0, std::ios::end);
  s.reserve(f.tellg());
  f.seekg(0, std::ios::beg);
  s.assign(std::istreambuf_iterator<char>(f), std::istreambuf_iterator<char>());
  return s;
}

inline int test_starting_utf8(char c) {
  if ((c & char(0b1110'0000)) == char(0b1100'0000))
    return 1;
  if ((c & char(0b1111'0000)) == char(0b1110'0000))
    return 2;
  if ((c & char(0b1111'1000)) == char(0b1111'0000))
    return 3;
  return -1;
}

inline bool test_trailing_utf8(char c) {
  return (c & char(0b1100'0000)) == char(0b1000'0000);
}

int main() {
  size_t nfiles = 0, nbytes = 0, nbom = 0, ascii_bom = 0, ascii_nobom = 0,
         utf8_bom = 0, utf8_nobom = 0, latin1_bom = 0, latin1_nobom = 0;
  for (auto &p : std::filesystem::recursive_directory_iterator("source")) {
    if (!std::filesystem::is_regular_file(p.path()))
      continue;
    nfiles++;
    if ((nfiles & (nfiles + 1)) == 0) {
      log(nfiles);
    }
    std::string data = slurp(p.path());
    nbytes += data.size();
```

```
  bool bom = (data.size() >= 3 && data[0] == char(0xEF) &&
              data[1] == char(0xBB) && data[2] == char(0xBF));

  const char *first = &data[0];
  const char *last = &data[data.size()];

  if (bom) {
    nbom++;
    first += 3;
  }

  bool ascii = true;
  bool utf8 = true;
  for (const char *p = first; p < last; p++) {
    if (!(*p & char(0x80)))
      continue;
    ascii = false;
    int ntrailing = test_starting_utf8(*p);
    if (ntrailing == -1) {
      utf8 = false;
      goto latin1;
    }
    for (int i = 0; i < ntrailing; i++) {
      p++;
      if (p == last || !test_trailing_utf8(*p)) {
        utf8 = false;
        goto latin1;
      }
    }
  }
latin1:
  if (ascii) {
    if (bom)
      ascii_bom++;
    else
      ascii_nobom++;
  } else if (utf8) {
    if (bom)
      utf8_bom++;
    else
      utf8_nobom++;
  } else {
    if (bom)
      latin1_bom++;
    else
      latin1_nobom++;
  }
}
log(nfiles);
```

```
  log(nbytes);
  log(nbom);
  log(ascii_bom);
  log(ascii_nobom);
  log(utf8_bom);
  log(utf8_nobom);
  log(latin1_bom);
  log(latin1_nobom);
}
```