Implementing foreign language support

Rolfe Bozier

5-Aug-2015

Agenda

- A bit of background
- What is internationalisation (I18N)?
- Supporting foreign languages
  - Encoding
  - Translation
  - Presentation

Background

- Goal: provide an application for use in other countries
  - Standalone software that is sold or distributed in another environment
  - Shared systems (“cloud”) that are used in multiple places
- Deployment has two phases:
  - Internationalisation (“I18n”)
  - Localisation (“L10n”)
- As software developers, we are mainly concerned about i18n

What are the issues?

- Language (content, collation, fonts, writing direction, …)
- Time-zone, currency
- Number/date/time formatting
- Identification issues (name/address/phone/titles/etc.)
- Cultural issues
- Paper size
- Public holidays
- Units of measure
- Map coordinate systems
- Regulatory issues (e.g. encryption, data retention)
- … many more

What are the issues?

- Language (content, collation, fonts, writing direction, …)
- Time-zone, currency
- Number/date/time formatting
- Identification issues (name/address/phone/titles/etc.)
- Cultural issues
- Paper size
- Public holidays
- Units of measure
- Map coordinate systems
- Regulatory issues (e.g. encryption, data retention)
- … many more

Language issues

- Three key aspects:
  - Defining and preserving the representation of data (encoding)
  - Conversion of text into the local language (translation)
  - Representation of translated text (fonts and UI)

Defining and preserving the representation of data (encoding)

Text encoding

- How do you store this text:
  - Привіт, я не росіянин.
- … so that it doesn’t come out like this:
  - ??????, ? ?? ????????
- This is not a trivial problem.

A brief history of text encodings

- 1960s-1980s – 7-bit ASCII
  - Great for English and maybe one or two other languages
  - 95 printable characters out of a possible 128
  - But at least every agreed on what characters were encoded as
    - Well, mostly

A brief history of text encodings

- 1990s – Let's use the 8th bit for more characters – ISO-8859

Encoding	Purpose
iso-8859-1 ("latin-1")	Western European languages
iso-8859-2 ("latin-2")	Central European languages
iso-8859-3 ("latin-3")	Southern European languages
iso-8859-4 ("latin-4")	Northern European languages
iso-8859-5	Cyrillic characters
iso-8859-6	Arabic characters
iso-8859-7	Modern Greek
iso-8859-8	Modern Hebrew
iso-8859-9 ("latin-5")	Like -1 but with Icelandic replaced with Turkish
iso-8859-10 ("latin-6")	Like -4, but with better support for Nordic languages
iso-8859-11	Thai
iso-8859-13 ("latin-7")	Like -4 and -10, but with better support for Baltic languages
iso-8859-14 ("latin-8")	Celtic languages
iso-8859-15 ("latin-9")	Like -1, but with more useful symbols
iso-8859-16 ("latin-10")	Better south-eastern European language support

A brief history of text encodings

- Several problems with this…
  - It only covers mostly European languages
  - What about languages with more than around 200 characters?
  - What about other symbols that might be useful?
  - What if you want to mix characters from multiple encodings?
  - How do you know which encoding is being used?

A brief history of text encodings

- Meanwhile, on the other side of the world…
- Asian languages need many more characters
  - Around 2,000 for Japanese
  - Up to 40,000 for Chinese
- Various encodings were developed

Encoding	Data size	Purpose
Big5	Double-byte	Traditional Chinese (Taiwan, Hong Kong, Macau)
GBnnnnn	Variable	Chinese (official)
Shift JIS	Variable (1-2 bytes)	Japanese (not well standardised)
EUC-JP	Variable	Japanese
EUC-CN	Variable	Chinese
EUC-KR	Variable	Korean
EUC-TW	Variable	Taiwan

A brief history of text encodings

- 1990s-present - Unicode to the rescue!
  - A modern, international standard for character definition and encoding
  - Identifies characters by abstract code points
  - Aims to cover all possible language encoding requirements
    - around 120,000 characters
    - around 129 modern/historic scripts
    - even scripts such as: Egyptian hieroglyphs, Linear A, Braille, Cherokee, Runic
    - private spaces: Klingon, Tengwar, Ferengi, ...
  - Also supports
    - combining characters
    - ligatures
    - emoticons
    - direction changing
    - mathematical and graphical symbols

A brief history of text encodings

- 1990s-present - Unicode to the rescue!
  - Space for around 1 million code points (entries)
  - All the characters you'll need are in the first 2^{16} entries
    - Also known as the Basic Multilingual Plane (BMP)
  - Defines several encodings that can be used:
    - UTF-8 (variable-length, 1-4 x 8-bit bytes)
    - UTF-16 (variable-length 1-2 x 16-bit entities)
    - UTF-32 (fixed-length 32-bit entities)

A brief history of text encodings

- 1990s-present - Unicode to the rescue!
  - Space for around 1 million code points (entries)
  - All the characters you'll need are in the first 2^{16} entries
    - Also known as the Basic Multilingual Plan (BMP)
  - Defines several encodings that can be used:
    - UTF-8 (variable-length, 1-4 x 8-bit bytes)
    - UTF-16 (variable-length 1-2 x 16-bit entities)
    - UTF-32 (fixed-length 32-bit entities)

HINT: choose this one!

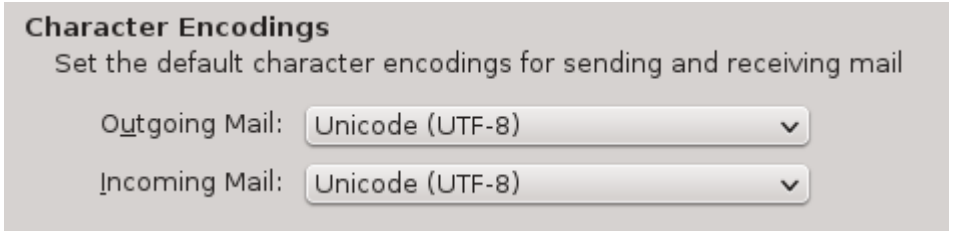
A brief history of text encodings

- Several problems with this…
  - It only covers European languages ✓
  - What about languages with more than around 200 characters? ✓
  - What about other symbols that might be useful? ✓
  - What if you want to mix characters from multiple encodings? ✓
  - How do you know which encoding is being used?

Only this problem remains…

Which encoding is being used?

- You generally can't tell just by looking at the text
- You need to make your choice and enforce it everywhere
- When you have the opportunity to specify what encoding is being used, then take it!
  - Database:
    - `CREATE DATABASE db_name CHARACTER SET utf8 …`
  - Web pages:
    - `<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />`
  - Email:



Implementation issues

- “Just use UTF-8 everywhere”
- Don’t convert between encodings unless you really know the source encoding
- C functions:
  - Don’t use strlen(), strchr(), strcmp()
    - If you think you need to use them, ask yourself why…
  - Do use strcoll()
  - Maybe use mb*() functions
  - Maybe use wc* functions

Conversion of text into the local language (translation)

Extracting text to be translated

- It is going to be painful
- Don’t just have message identifiers in your code
  - At least have the English text as a comment nearby
- Automate wherever possible
- You’re going to have to re-write some of your text / code…

Extracting text to be translated

- printf() and ordering of the replacements

```
printf
(
    ”You need to specify the correct %s mode for %s paper\n”,
    mode_str,
    paper_name
);
```

What if these words should be in a different order?

Extracting text to be translated

- Word construction

```
printf
(
    ”%d page%s printed\n”,
    n,
    n == 1 ? ”” : ”s”
);
```

That only works for English…

Extracting text to be translated

- Short phrases or jargon

```
printf
(
    ”file: %s\n”,
    result
);
```

Should the translator treat “file” as a noun, verb, adjective, or what?

Testing text translation

- If it's not continually tested, it will break
  - Mishandling of encoding can result in bad output
  - Some text is always missed out
  - You won't find out until late in the process… too late
- You need a way to verify translation during development…

Testing text translation

- If it's not continually tested, it will break
  - Mishandling of encoding can result in bad output
  - Some text is always missed out
  - You won't find out until late in the process… too late
- You need a way to verify translation during development…
- Automatic translation to "jive"

  Please enter a valid printer name
  "Please enta' some valid printa' dojigger. What it is, Mama. Right On!"

  - Amusing, somewhat readable, doesn't test non-7-bit data
  - **Not** for demonstrations in the southern US states

Testing text translation

- If it’s not continually tested, it will break
  - Mishandling of encoding can result in bad output
  - Some text is always missed out
  - You won’t find out until late in the process… too late
- You need a way to verify translation during development…
- Custom character set

    Printer is not known
    “Ṗṝȉńŧëṝ ȉṧ ṋö̃ŧ ḵńö̃ẅń”

  - Solves most of the above issues

Actual text translation

- Send your message files to a professional translation agency
  - Preferably one with experience in technical translation
  - They should be capable of raising issues with ambiguity or jargon
  - Not the receptionist or a friend…
- Double-check the translation with someone else
  - Maybe a customer or an employee

What could possibly go wrong?

What could possibly go wrong?

"I am not in the office at the moment. Send any work to be translated"

Representation of translated text (fonts and UI)

Presentation issues

- Obviously you need the right fonts available
  - Usually this is managed at the user's end
- Be aware that the translated text may not be a drop-in replacement for the English version

Presentation issues

- Obviously you need the right fonts available
  - Usually this is managed at the user's end
- Be aware that the translated text may not be a drop-in replacement for the English version
  - For example:

Submit problem details:

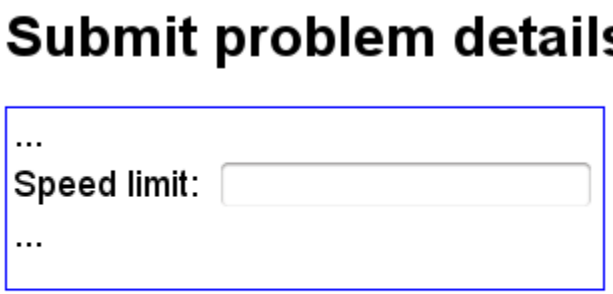
...
Speed limit:
...

Submit problem details:

...
制限速度:
...

Presentation issues

- Obviously you need the right fonts available
  - Usually this is managed at the user's end
- Be aware that the translated text may not be a drop-in replacement for the English version
  - For example:



Presentation issues

- Obviously you need the right fonts available
  - Usually this is managed at the user's end
- Be aware that the translated text may not be a drop-in replacement for the English version
  - For example:

Submit problem details:

...
Speed limit:
...

Submit problem details:

...
الحد الأقصى للسرعة:
...

Recap

- Language (content, collation, fonts, writing direction, …)
- Time-zone, currency
- Number/date/time formatting
- Identification issues (name/address/phone/titles/etc.)
- Cultural issues
- Paper size
- Public holidays
- Units of measure
- Map coordinate systems
- Regulatory issues (e.g. encryption, data retention)
- … many more

We didn't even consider these other issues…

Take away messages

1. You will need to identify and overcome your preconceptions
2. You need to be disciplined in your development processes
3. You will need outside help
4. By the time you find your mistakes, it could be too late…