

Internationalization: Multi-language Support

Presented by Chao Gao



Questions

- **What is internationalization?**
 - **Why do we need internationalization for Linux or software?**
 - **How much do you know about internationalization encoding?**
 - **How did you get those information?**
-



Internationalization and Localization

- **Internationalization** (i18n) is the process of designing a software application so that it can be adapted to various languages and regions without engineering changes.
- **Localization** (L10n) is the process of adapting software for a specific region or language by adding locale-specific components and translation text.



Internationalization and Localization

■ Practice

- Place text in resource strings which are loaded during program execution as needed
- These strings, stored in resource files, are relatively easy to translate
- Programs are often built to reference resource libraries depending on the selected locale data



Some Definitions

- **Character repertoire** is the full set of abstract characters that a system supports.
- A **coded character set** specifies how to represent a repertoire of characters using a number of non-negative integer codes called code points.



Character Sets

- **Character sets affect two fundamental parts of your code:**
 - How you store or transmit data, your file format
 - String processing, the logic with which you manipulate text
- **Character sets do not solve:**
 - Locale-awareness, formatting preferences
 - Special input requirements, keyboard layouts, IMEs
 - Text layout, fonts and other display issues



Unicode

- An industry standard allowing computers to consistently represent and manipulate text expressed in most of the world's writing systems
- Consists of a repertoire of about 100,000 characters, a set of code charts for visual reference, an encoding methodology and set of standard character encodings, an enumeration of character properties.....



Unicode (cont)

- Takes the role of providing a unique code point – a number, not a glyph – for each character
- Represents a character in an abstract way and leaves the visual rendering (size, shape, font or style) to other software, such as a web browser or word processor

UCS Implementation Levels

A combining character is not a full character by itself. It is an accent or other diacritical mark that is added to the previous character.

Level 1

Combining characters and Hangul Jamo characters are not supported. [Hangul Jamo are an alternative representation of precomposed modern Hangul syllables as a sequence of consonants and vowels. They are required to fully support the Korean script including Middle Korean.]

Level 2

Like level 1, however in some scripts, a fixed list of combining characters is now allowed (e.g., for Hebrew, Arabic, Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugo, Kannada, Malayalam, Thai and Lao). These scripts cannot be represented adequately in UCS without support for at least certain combining characters.

Level 3

All UCS characters are supported, such that, for example, mathematicians can place a tilde or an arrow (or both) on any character.

Example



0041;LATIN CAPITAL LETTER A;Lu;0;L;;;;;N;;;;;0061;

Representative
glyph

A

Semantic
properties

Code point: 0041

Name: LATIN CAPITAL LETTER A

General category: Uppercase letter (Lu)

Canonical combining class: Standard spacing (0)

Bidirectional category: Left-to-right (L)

Mirrored: no (N)

Lowercase mapping: 0061

More Examples (chinese)

U+97F3 CJK UNIFIED IDEOGRAPH-97F3





Encodings For Unicode: UTF-8

- A variable-length character encoding for Unicode
- Backwards compatible with ASCII
- The preferred encoding for e-mail, web pages, and other places where characters are stored or streamed.



UTF-8 Properties

- UCS characters U+0000 to U+007F (ASCII) are encoded simply as bytes 0x00 to 0x7F (ASCII compatibility). This means that files and strings which contain only 7-bit ASCII characters have the same encoding under both ASCII and UTF-8.
- All UCS characters >U+007F are encoded as a sequence of several bytes, each of which has the most significant bit set. Therefore, no ASCII byte (0x00-0x7F) can appear as part of any other character.



UTF-8 Properties (Cont)

- The first byte of a multibyte sequence that represents a non-ASCII character is always in the range 0xC0 to 0xFD and it indicates how many bytes follow for this character. All further bytes in a multibyte sequence are in the range 0x80 to 0xBF. This allows easy resynchronization and makes the encoding stateless and robust against missing bytes.
- All possible 2^{31} UCS codes can be encoded.
- UTF-8 encoded characters may theoretically be up to six bytes long, however 16-bit BMP characters are only up to three bytes long.
- The bytes 0xFE and 0xFF are never used in the UTF-8 encoding.

UTF-8 (cont)

UTF-8	Serialized Bytes					
Unicode Range	1 st	2 nd	3 rd	4 th	5 th	6 th
U-00000000 - U-0000007F	0nnnnnnn					
U-00000080 - U-000007FF	110nnnnn	10nnnnnn				
U-00000800 - U-0000FFFF	1110nnnn	10nnnnnn	10nnnnnn			
U-00010000 - U-001FFFFF	11110nnn	10nnnnnn	10nnnnnn	10nnnnnn		
U-00200000 - U-03FFFFFF	111110nn	10nnnnnn	10nnnnnn	10nnnnnn	10nnnnnn	
U-04000000 - U-7FFFFFFF	1111110n	10nnnnnn	10nnnnnn	10nnnnnn	10nnnnnn	10nnnnnn

Example

The Cent Sign (¢), which is Unicode U+00A2, is encoded into UTF-8 in this way:

- In the range of U+0080 to U+07FF → 110yyyyy 10zzzzzz
- Hexadecimal 0x00A2 is equivalent to binary → 0000 0000 1010 0010
- Put in their order into the positions marked by "y"-s and "z"-s → 110**00010** 10**100010**
- The final result is the two bytes, expressed as the two hexadecimal bytes 0xC2 0xA2



Other Character Encoding

- UCS-2, UCS-2BE, UCS-2LE, UCS-4, UCS-4LE, UCS-4BE
- UTF-7, UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF-32, UTF-32BE, UTF-32LE



Canonical equivalence

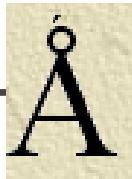
- Canonical equivalence is a narrower form of equivalence that preserves visually and functionally equivalent characters.

For example, precomposed diacritic letters are considered canonically equivalent to their decomposed letter and combining diacritic marks.

$$\text{'ü'} = \text{'u'} + \text{'̂'}$$

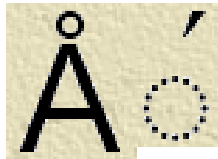
Similarly, Unicode unifies several Greek diacritics and punctuation characters that have the same appearance to other diacritics.

Canonical equivalence



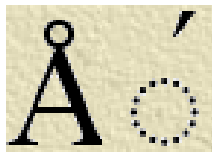
01FA

LATIN CAPITAL LETTER A WITH RING ABOVE AND ACUTE



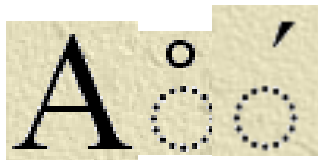
212B 0301

ANGSTROM SIGN
COMBINING ACUTE ACCENT



00C5 0301

LATIN CAPITAL LETTER A WITH RING ABOVE
COMBINING ACUTE ACCENT



0041 030A 0301

LATIN CAPITAL LETTER A
COMBINING RING ABOVE
COMBINING ACUTE ACCENT

Wide/Multi-bytes Characters



- Asian languages have repertoires of thousands of ideographic characters; normally two octets are used per character, and two per glyph.
- There are two formats for such large encodings:
 - **wide character format:** each character is represented by the same number of octets.
 - **multi-byte character format:** different characters may be represented by different numbers of octets.

UTF-8 in Operating Systems



- Windows uses Unicode as the sole internal character encoding.
- The Java and .NET bytecode environments, Mac OS X, and KDE also use it for internal representation
- UTF-8 has become the main storage encoding on most Unix-like operating systems



Using UTF-8 in Linux

- Most newer distributions have standardized on UTF-8
RedHat 8.0, SuSE 8.1, etc.
- Most applications only have a Level 1 implementation
Terminals, output, etc.
- Need to set your locale to a UTF-8 locale
 - unless a recent Linux distribution, you are still very likely using a legacy locale based on ISO-8859 or other national encoding.
 - If you are using some UNIX-based OS other than Linux, it is even less likely that you are already using a UTF-8 locale.
 - To determine your current locale settings, refer to:
<http://eyegene.ophthy.med.umich.edu/unicode/>

<http://www.cl.cam.ac.uk/~mgk25/unicode.html#linux>

C support for Unicode and UTF-8

- Call this program with the locale setting LANG=de_DE and the output will be in ISO 8859-1.
- Call it with LANG=de_DE.UTF-8 and the output will be in UTF-8.
- The %ls format specifies in *printf* calls *wcsrtombs* in order to convert the wide character argument string into the locale-dependent multi-byte encoding.

```
#include <stdio.h>
#include <locale.h>

int main()
{
    if (!setlocale(LC_CTYPE, "")) {
        fprintf(stderr, "Can't set the specified locale! "
                "Check LANG, LC_CTYPE, LC_ALL.\n");
        return 1;
    }
    printf("%ls\n", L"Schöne Grüße");
    return 0;
}
```

You can query the name of the character encoding in your current locale with the command *locale charmap*. This should say UTF-8 if you successfully picked a UTF-8 locale in the LC_CTYPE category. The command *locale -m* provides a list with the names of all installed character encodings.



Internationalization of Linux

- **Linux is generally available in many languages**
 - The Debian distribution currently loads over 200 languages
 - Ubuntu's LiveCD allows you to conveniently pick to load any and all choices you want from dozens of languages when you boot up or when you choose System-> Administration-> Language Support
 - Ubuntu's 7.04 version covers languages from Afar to Zulu including dialect choices such as Ancient or Modern Greek or several varieties of English



Terminal Emulation and communication

- xterm as shipped with XFree86 4.0 or higher works correctly in UTF-8 locales if you use an *-iso10646-1 font
- C-Kermit has supported UTF-8 as the transfer, terminal, and file character set since version 7.0.
- mlterm is a multi-lingual terminal emulator that supports UTF-8 among many other encodings, combining characters, XIM.
- Edmund Grimley Evans extended the BOGL Linux framebuffer graphics library with UCS font support and built a simple UTF-8 console terminal emulator called bterm with it.
- Uterm purports to be a UTF-8 terminal emulator for the Linux framebuffer console.
- Pluto, Juliusz Chroboczek's paranormal Unicode converter, can guess which encoding is being used in a terminal session, and converts it on-the-fly to UTF-8.

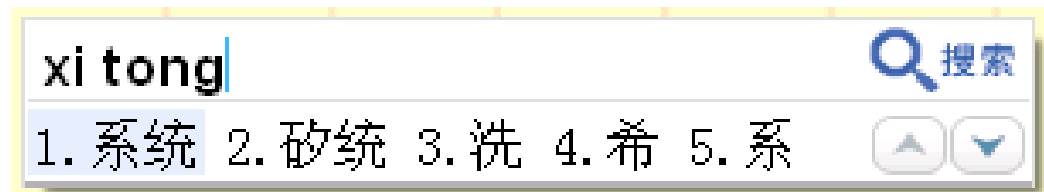
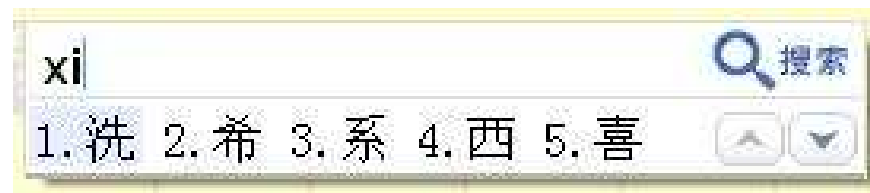


UTF-8 in Input Methods

- **Basic method:** a beginning sequence is followed by the hexadecimal representation of the codepoint and the ending sequence
- **Screen-selection entry method:** characters are listed in a table in a screen, such as with a character map program

Chinese Input Example

系统





UTF-8 in Email

- For e-mail transmission of Unicode the UTF-8 character set and the Base64 transfer encoding are recommended
- For an email message, you are expected to have a string in the header of the form Content-Type: text/plain; charset="UTF-8"



UTF-8 in Email

- The Mutt email client has worked since version 1.3.24 in UTF-8 locales.
- Exmh is a GUI frontend for the MH or nmh mail system and partially supports Unicode starting with version 2.1.1 if Tcl/Tk 8.3.3 or newer is used.
- The popular Pine email client lacks UTF-8 support and is no longer maintained. Switch to its successor Alpine, a complete reimplementaion by the same authors, which has excellent UTF-8 support.



UTF-8 in Web

- **All W3C recommendations have used Unicode as their document character set since HTML 4.0**

```
<html>
```

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html;  
charset=utf-8">
```

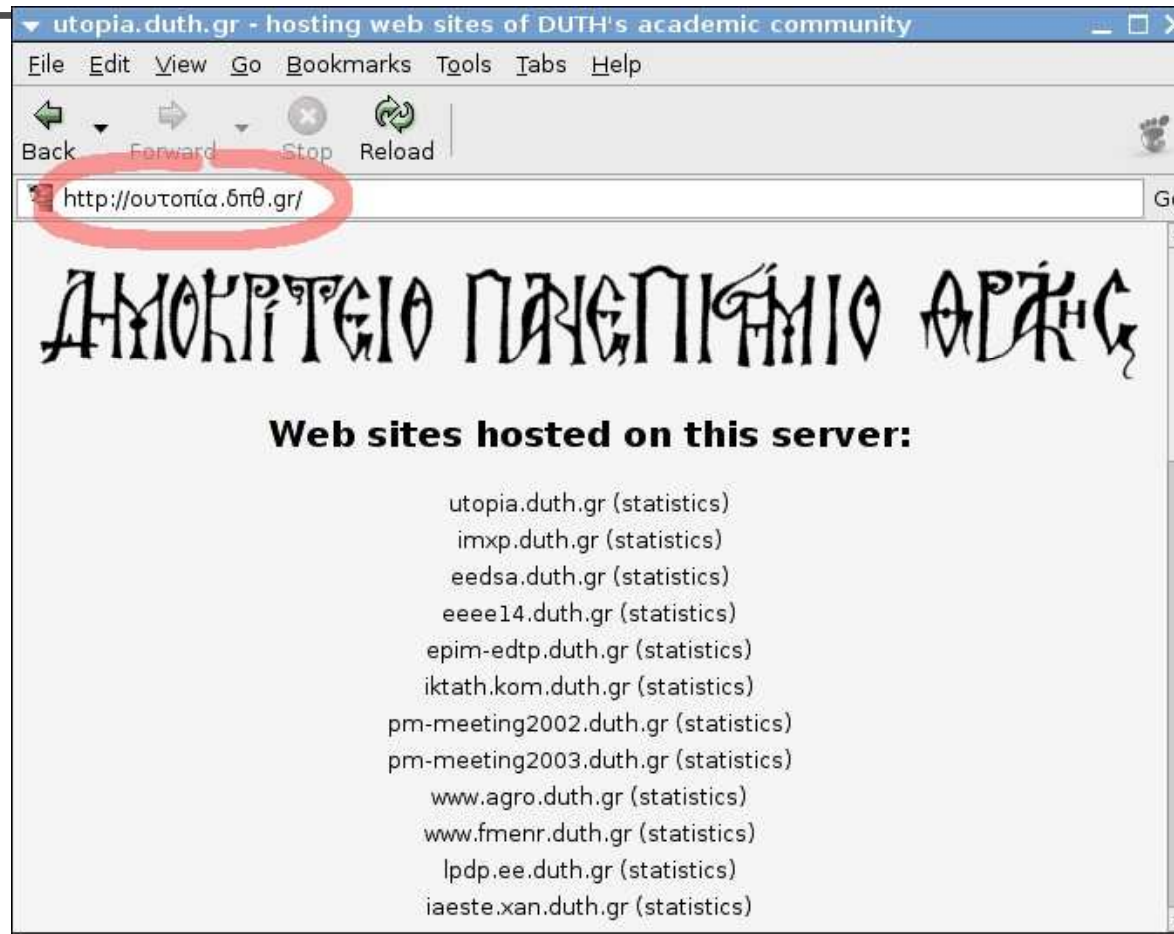
- **Most modern web browsers such as Mozilla Firefox have pretty decent UTF-8 support today**

Internationalized Domain Name



- An **internationalized domain name** (IDN) is an Internet domain name that (potentially) contains non-ASCII characters
- Non-ASCII domain names should be converted to a suitable ASCII-based form by web browsers and other user applications
- **Internationalizing Domain Names in Applications** (IDNA) is a mechanism defined in 2003 for handling internationalized domain names containing non-ASCII characters
- IDNA was designed for maximum backward compatibility with the existing DNS system, which was designed for use with names using only a subset of the ASCII character set.

Internationalized Domain Name (Example)



http://en.wikipedia.org/wiki/Internationalized_domain_name

Internationalized Domain Name

- The conversions between ASCII and non-ASCII forms of a domain name are accomplished by algorithms called **ToASCII** and **ToUnicode**.

- Example of IDNA encoding :

Bücher.ch (“Bücher” is German for “books”, and .ch is the country domain for Switzerland)

Bücher → Nameprep(case-folding to lowercase and removal of some generally invisible code points) → bücher → Punycode → bcher-kva → ACE prefix → xn--bcher-kva → xn--bcher-kva.ch

Note: the ToASCII algorithm can fail in a number of ways; for example, the final string could exceed the 63-character limit for the DNS.

- ToUnicode reverses the action of ToASCII

What programming languages support Unicode?

- Have special data types for Unicode (after around 1993)
 - Ada95, Java, TCL, Perl, Python, C# and others.
- ISO C 90
 - Specifies mechanisms to handle multi-byte encoding and wide characters.
 - The type *wchar_t*, usually a signed 32-bit integer, can be used to hold Unicode characters.
- ISO C 99
 - Some problems with backwards compatibility.
 - The C compiler can signal to an application that *wchar_t* is guaranteed to hold UCS values in all locales.



Conclusion

- Unicode solves the problem of multi-language support and compatibles to the 7-bit ASCII.
 - UTF-8 has been widely used in operating systems, email, web.....
 - Encoding each Chinese character(other east Asian languages as well) in Unicode may not be a efficient method
-



Recommended Reading

Markus Kuhn's famous ``UTF-8 and Unicode FAQ for Unix/Linux'' pages (**Highly recommended**)

<http://www.cl.cam.ac.uk/~mgk25/unicode.html>

Bruno Haible's ``Unicode HOWTO''

<ftp://ftp.ilog.fr/pub/Users/haible/utf8/Unicode-HOWTO.html>
