Character Sets

Original by Jarret Raim, Spring 2004
Updated and expanded, 2007-2013

Characters, bytes, storage & display

- We've talked about characters and strings
- We know how to
  - Find out how much storage they require
  - Write them to a file
  - Write them to the screen

Characters, bytes, storage & display

- What happens when you want to write mathematical symbols like ⊨,√,∞, and ∑
- Or characters from other languages, such as غقĖ,Б,ć, and ö
- Such characters are outside the ASCII codes, and may require more screen space to print and more storage space

Definitions

- Character Repertoire
  - A set of characters where no internal presentation in computers or data transfer is assumed.
  - Does not define an ordering for the characters.
  - Usually defined by specifying names of characters and a sample (or reference) presentation of characters in visible form.

Definitions

- Character Code
  - Defines a one-to-one correspondence between characters in a repertoire and a set of nonnegative integers, called a code position.
  - Aka: *code number*, *code value*, *code element*, *code point*, *code set value* - and just *code*.
  - Note: The set of nonnegative integers corresponding to characters need not consist of consecutive numbers.

Definitions

- Character Encoding
  - A method (algorithm) for presenting characters in digital form by mapping sequences of code numbers of characters into sequences of octets.
  - In the simplest case, each character is mapped to an integer in the range 0 - 255 according to a character code and these are used as such as octets.
  - Eg: 7-bit ASCII, 8-bit ASCII, UCS, Unicode, UTF-6, UTF-16, etc.

ASCII & Friends

- The original ASCII is a 7-bit encoding using 0-127 to define basic US characters
- ISO Latin 1 is ASCII with European characters. (8-Bit)
- Contain control codes as well as text.

ASCII value	Character	Control character
000	(null)	NUL
001	☺	SOH
002	☻	STX
003	♥	ETX
004	♦	EOT
005	♣	ENQ
006	♠	ACK
007	(beep)	BEL
008	◘	BS
009	(tab)	HT
010	(line feed)	LF
011	(home)	VT
012	(form feed)	FF
013	(carriage return)	CR
014	♫	SO
015	☼	SI
016	►	DLE
017	◄	DC1
018	↕	DC2
019	!!	DC3
020	π	DC4
021	§	NAK

More ASCII Love

- Even basic 7-bit ASCII is not safe
  - Many “national variants” of ASCII replace some characters with international ones.
- Safe ASCII Characters
  - 0-9
  - A-Z and a-z
  - ! " % & ' ( ) * + , - . / : ; < = > ?
  - Space

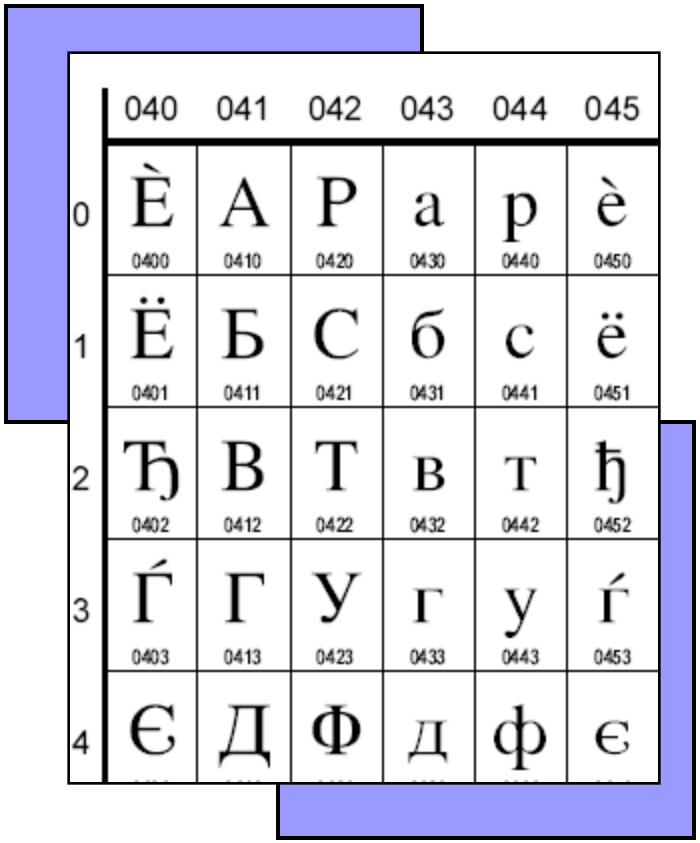
glyph	official Unicode name	National variants
#	NUMBER SIGN	£ Ù
$	DOLLAR SIGN	¤
@	COMMERCIAL AT	É § Ä à ³
[	LEFT SQUARE BRACKET	Ä Æ ° â ¡ ÿ é
\	REVERSE SOLIDUS	Ö Ø ç Ñ ½ ¥
]	RIGHT SQUARE BRACKET	Å Ü § ê é ¿ |
^	CIRCUMFLEX ACCENT	Ü î
_	LOW LINE	è
`	GRAVE ACCENT	é ä µ ô ù
{	LEFT CURLY BRACKET	ä æ é à ° ¨
|	VERTICAL LINE	ö ø ù ò ñ ƒ
}	RIGHT CURLY BRACKET	å ü è ç ¼
~	TILDE	ü ¯ ß ¨ û ì ´ _

Other Ridiculousness

- Other 8-Bit ASCII Extensions
  - DOS Code Pages
  - Macintosh Character Codes
  - IBM's EBCDIC (Mainframes)
- Windows did not conform to any known standards until NT switched over to using Unicode encoded as UTF-16.

The Solution: Unicode

- Unicode is a practical description of the ISO 10646 standard known as UCS or the Universal Character Set.
- Up to 1,112,064 characters can be encoded.
- As of Oct 2010, there were >109K characters (Unicode 6.0).
- The encoding is not defined
  - Several implementations



Encodings For Unicode

- Most Common: UTF-8
  - Character codes less than 128 (effectively, the ASCII repertoire) are presented "as such", using one octet for each code.
  - All codes with the high bit set to 1 (i.e., not ASCII) link to a mechanism for rendering Unicode characters with up to 6 octets.
  - Allows space savings and compatibility at the cost of implementation complexity.

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

Unicode Complexity

- Characters can be encoded multiple ways.
  - Π is encoded as:
    - GREEK_CAPITAL_LETTER_PI
    - N_ARY_PRODUCT
  - Ä can be encoded as:
    - LATIN CAPITAL LETTER A WITH DIAERESIS
    - The symbol A with a link to the umlaut diacritic
- All characters can be represented by the U+nnnn notation.
- Other Implementations
  - UCS-2, UCS-2BE, UCS-2LE, UCS-4, UCS-4LE, UCS-4BE, UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF-32, UTF-32BE, UTF-32LE

Unicode Implementation

- Level 1
  - Combining characters and Hangul Jamo characters are not supported. [Hangul Jamo are required to fully support the Korean script including Middle Korean.]
- Level 2
  - Like level 1, except limited combining characters are supported for some languages.
- Level 3
  - All UCS characters are supported, such that, for example, mathematicians can place a tilde or an arrow (or both) on any character.

Programming Languages

- Special Data Types for Unicode
  - 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.

Using Unicode in Linux

- Most distributions have standardized on UTF-8.
- Good performance requires hand tuning for UTF-8
  - Grep without hand tuning was 100x slower in multi-byte mode than in single-byte mode.

Using Unicode

- Libraries must support Unicode formats.
- New strlen() definitions:
  - Number of bytes (UTF-8 can still use regular strlen())
  - Number of characters
  - Display width (# of cursor positions)
- Examples
  - u8_strlen(), u8_printf()
- Application must pay attention to the locale setting for UTF-8 activation.

Unicode Functions

```
char *setlocale (int          category,
                 const char *locale)
```

- Setting the locale
  - setlocale (LC_NUMERIC, "Germany");
- Defines all numbers returned from libc to use German notation.

```
#include <conio.h>

int     gettext(int _left, int _top, int _right, int _bottom,
                void *_destin);
```

- gettext() returns the translations of strings in a message database.

```
/* get the translation for a string
   corresponding to a greeting
   (perhaps "Hello world!") */
printf(gettext("greeting"));
```

- No more reliance on the underlying numerical representation of ASCII.

BAD: l = c - 'A' + 'a';

GOOD: l = tolower(c);

Supporting i18n in C

- There are many new macros and system calls to support wide and multi-byte characters

```
wprintf(L"%1$d:%2$.*3$d:%4$.*3$d\n",
hour, min, precision, sec);
```

- See wide.c for another example.

Unicode on the Web

- Should be specified in a MIME header for **ALL** communications internal and external.
  - The header is sent in ASCII (UTF-8)

```
X-Mailer: Mozilla 4.0 [en] (Win95; I) MIME-Version: 1.0
To: someone@cs.tut.fi
Subject: Test
X-Priority: 3 (Normal)
Content-Type: text/plain; charset=x-UNICODE-2-0-UTF-7
Content-Transfer-Encoding: 7bit

HTTP/1.1 200 OK
Date: Wed, 25 Apr 2007 01:23:32 GMT
Server: Apache/2.0.54 (Fedora)
Last-Modified: Wed, 04 Apr 2007 15:03:37 GMT
ETag: "242873-1b55-c2456440"
Content-Length: 6997
Connection: close
Content-Type: text/html; charset=UTF-8
```

Unicode in DNS

- DNS only supports ASCII domain names.
- IDNA is a system to allow international characters to be used in domains names.
  - Converts Unicode characters into ASCII for DNS.
  - In 2010, >20 countries requested IDN ccTLDs (including in Arabic, Chinese, Russian & Thai)
  - A number have been approved; some are operational (Egypt, Saudi Arabia, Russia, etc.)

Conclusions

- Unicode is complex, but using UTF-8 allows a programmer to get many of the benefits of internationalization for free.
- Using STL data structures and other Unicode aware libraries will significantly reduce the pain of using Unicode (kiss char* goodbye).
- Assume that there will be difficulties with internationalization.

Sources

- Quick Overview
  - http://www.linuxjournal.com/article/3327
- Sun's Internationalization Reference
  - http://developers.sun.com/dev/gadc/educationtutorial/creference/
- Programming for Internationalization FAQ
  - http://faqs.cs.uu.nl/na-dir/internationalization/programming-faq.html
- UTF-8 and Unicode FAQ
  - http://www.cl.cam.ac.uk/~mgk25/unicode.html
- Plus many more.