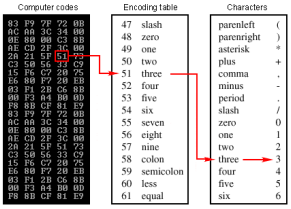
Using Unicode and Other Encodings in your Programs

Jeroen Pluimers
better office benelux
jpluimers@better-office.com

Note

- This is an overview session
  - It explains about Unicode and Encodings
  - It points to lots of references for further Delphi and Unicode reading
  - It has very few demos
  - But it does some show pitfalls

ASCII - Encoding - Display



- This is what we have been doing forever
- Bytes were equal to characters:
  - C := Char(B);

Picture courtesy of http://www.sketchpad.net/macwinfontsintro1.htm

Data – Encoding – Display

ASCII
CP 437
Windows 1252
UTF-8
…

Pictures courtesy of http://datalib.ed.ac.uk/ and http://www.ascenderfonts.com/info/simplified-chinese-fonts.aspx

Some encodings

- Single byte (8-bit)
  - ASCII — aka ISO 646 (actually 7-bit: only 0-127 are defined)
  - EBCDIC — aka CP390 – if you do AS/400, iSeries, OS390/Mainframes
  - ISO 8859 — split into 'regions' as …-1 through …-15, except …-12
  - DOS Code Pages — most pupular: CP437, CP850, CP852, CP858
  - Windows Code Pages — most popular: Windows-1250, Windows-1252
- Two byte (16-bits)
  - ISO-2022 — for …JP, …-CN, …-KR (kinds of 7-bit multi byte encoding)
  - EUC — for …JP, …-CN, …-KR (kinds of 8-bit multi byte encoding)
- Multi byte (8 ,16 or 32-bits)
  - Unicode — UTF-8, UTF-16, UTF-32
  - GB 18030 — Chinese; mapable to Unicode
- Wikipedia has tables for most 8-bit encodings
- UTF8 encodings tables: http://www.utf8-chartable.de
- See: http://en.wikipedia.org/wiki/Character_encoding
  http://en.wikipedia.org/wiki/Code_page
  http://en.wikipedia.org/wiki/Multi-byte_character_set
  http://en.wikipedia.org/wiki/CJK_characters
  http://en.wikipedia.org/wiki/Comparison_of_Unicode_encodings

Unicode fonts

- No font can have all Unicode code points
  - Unicode defines 100k+ code points
  - Fonts have a maximum of 65k code points
- Some fonts having many Unicode code points
  - GNU Unifont — http://unifoundry.com/unifont.html
  - Code2000 — http://www.code2000.net/code2000_page.htm
  - New Gulim / GulimChe — http://r.office.microsoft.com/r/rlidDownloadDetail?p1=2002&p2=ie_ko
  - Arial Unicode MS — included since MS Office 2000
  - Everson Mono — http://www.evertype.com/emono/
- Notable font
  - Lucida Sans Unicode – first ever font supporting Unicode
- See: http://en.wikipedia.org/wiki/Unicode_typefaces

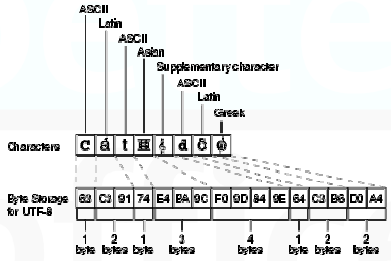
Explanation of encodings

- Encoding methods as falling into two main categories:
  - The Old Way:
    Each character has a specific and direct representation on a computer.
  - The New Way:
    Each character is a concept, and can be represented in multiple ways.
- The new way is what Unicode does:
  - it maps every letter of every language to a unique number (called a code point),
  - so an upper-case "D" in Unicode maps to U+0044 (aka U-00000044)
  - for UTF-8, encoding of these code points is done like this:

U-00000000 – U-0000007F:	0xxxxxxx
U-00000080 – U-000007FF:	110xxxxx 10xxxxxx
U-00000800 – U-0000FFFF:	1110xxxx 10xxxxxx 10xxxxxx
U-00010000 – U-001FFFFF:	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
U-00200000 – U-03FFFFFF:	111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
U-04000000 – U-7FFFFFFF:	1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

  - See: http://danielmiessler.com/study/encoding/

Mostly used: UTF-8 encoding



Picture courtesy of
http://stanford.edu/dept/itss/docs/oracle/10g/server.101/b10749/ch2charset.htm

Read these

- Marco Cantu
  - Whitepaper "Delphi and Unicode"
    New
    http://etnaweb04.embarcadero.com/resources/technical_papers/Delphi-and-Unicode_Marco-Cantu.pdf
    Old
    http://www.codegear.com/article/38548/images/38548/Delphi_and_Unicode.pdf
  - Presentation "Delphi and Unicode"
    http://it-republik.de/konferenzen/delphi_live/material/DelpiLive09_Cantu_Unicode.pdf
- Nick Hodges
  - Whitepaper "Delphi in a Unicode World"
    http://etnaweb04.embarcadero.com/resources/technical_papers/Delphi%20in%20a%20Unicode%20World_updated.pdf

On-line help: topics

- What's New in Delphi and C++Builder 2009
  - ms-help://embarcadero.rs2009/devcommon/whatsnewtiburon_xml.html
    http://edn.embarcadero.com/article/38869
- String Types
  - Base: ShortString, AnsiString, WideString, UnicodeString
    - http://docs.embarcadero.com/products/rad_studio/delphiAndcpp2009/HelpUpdate2/EN/html/devcommon/stringtypes_xml.html
  - Unicode (including internal memory layouts of the string types)
    - http://docs.embarcadero.com/products/rad_studio/delphiAndcpp2009/HelpUpdate2/EN/html/devcommon/unicodeinide_xml.html
  - Enabling for Unicode
    - http://docs.embarcadero.com/products/rad_studio/delphiAndcpp2009/HelpUpdate2/EN/html/devcommon/enablingunicode_xml.html
  - RawByteString (AnsiString with CodePage $ffff)
    - http://docs.embarcadero.com/products/rad_studio/delphiAndcpp2009/HelpUpdate2/EN/html/delphivclwin32/System_RawByteString.html
  - UTF8String (AnsiString with CodePage 65001)
    - http://docs.embarcadero.com/products/rad_studio/delphiAndcpp2009/HelpUpdate2/EN/html/delphivclwin32/System_UTF8String.html
- String <> PChar conversions
  - PChar fundamentals
    - http://docs.embarcadero.com/products/rad_studio/delphiAndcpp2009/HelpUpdate2/EN/html/devwin32/stringdependencies_xml.html
  - Returning a PChar Local Variable
    - http://docs.embarcadero.com/products/rad_studio/delphiAndcpp2009/HelpUpdate2/EN/html/devwin32/passingalocalvariableaspchar_xml.html
  - Passing a Local Variable as a PChar
    - http://docs.embarcadero.com/products/rad_studio/delphiAndcpp2009/HelpUpdate2/EN/html/devwin32/returningapcharlocalvariable_xml.html

Unchanged since D2009

• AnsiChar tkChar	PAnsiChar none	AnsiString tkLString
• WideChar tkWChar	PWideChar none	WideString tkWString
• ShortString tkString		

  - ShortString and WideString are NOT reference counted
  - AnsiString (and UnicodeString) ARE reference counted
  - There is no UnicodeShortString

Changed since D2009

- Pre D2009
  - Char = AnsiChar
  - PChar = PAnsiChar
  - String = AnsiString
  - tkString (TypeInfo)
- D2009+
  - Char = UnicodeChar
  - PChar = PUnicodeChar
  - String = UnicodeString
  - tkUString (TypeInfo)

-8	RefCount
-4	Length
0	String content

-12	Code Page
-10	Element Size
-8	RefCount
-4	Length
0	String content

New since D2009

- AnsiString
  - can have CodePage, i.e. AnsiString(1252) for CodePage 1252
  - CodePage is same as CodePage of the running system (NOT the compiling system)
  - RawByteString = AnsiString($ffff)
  - UTF8String = AnsiString(65001)
- TypeInfo for tkLString now has CodePage field
- UnicodeString
  - Internally uses UTF-16 (like WideString)
- Implicit conversion from
  - AnsiString(CodePage1) to AnsiString(CodePage2)
  - AnsiString to UnicodeString
- Classes
  - TCharacter (Characters)
  - TEncoding (SysUtils)
  - TStringBuilder (SysUtils)
  - TTextReader (Classes)
    - TStreamReader
    - TStringReader
  - TTextWriter (Classes)
    - TStreamWriter
    - TStringWriter
- Units
  - Character
  - AnsiStrings
- CodePage list:
  - http://msdn.microsoft.com/en-us/library/dd317756(VS.85).aspx

CodePages used internally

- "Code Page Identifiers" http://msdn.microsoft.com/en-us/library/dd317756(VS.85).aspx

```
// some consts that are internal to the System unit
const
  CP_UTF8 = 65001; // UTF8String CodePage
  CP_UTF16 = 1200; // string CodePage

// some literals that are internal to the System unit
const
  CP_ACP = 0; // default AnsiString CodePage
  CP_RawByteString = $ffff; // RawByteString CodePage
```

Some things to watch for

String constants don't always have the right CodePage

- http://qc.embarcadero.com/wc/qcmain.aspx?d=69517

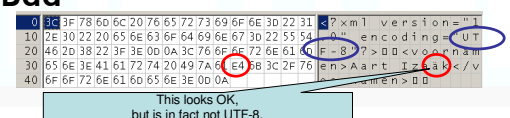
```
const
  CP_VietnameseString = 1258;
type
  VietnameseString = type
   AnsiString(CP_VietnameseString);
// these literal string consts
// do not have the right CodePage
// they all get 1252 assigned:
const
  LF_UTF8String = UTF8String(#10);
  LF_RawByteString = RawByteString(#10);
  LF_VietnameseString = VietnameseString(#10);
```

When converting by hand:

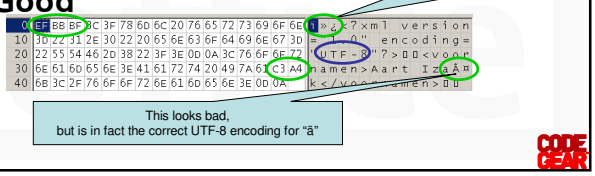
```
function ReadMessageBufferLen(IOHandler: TIdIOHandler;
   ABytes: Integer):
  RawByteString;
{$IF Declared(RTLVersion) and (RTLVersion >= 20.0)}
var
  Bytes: TBytes;
{$IFEND}
begin
{$IF Declared(RTLVersion) and (RTLVersion >= 20.0)}
  IOHandler.ReadBytes(Bytes, ABytes);
  SetLength(Result, Length(Bytes));
  Move(Bytes[0], Result[1], ABytes);
{$ELSE}
  Result := IOHandler.ReadString(ABytes, Indy8BitEncoding);
{$IFEND}
end;
```

Detecting wrong encodings

- Bad



- Good



Reencoding files

- Two examples:
  - Utf82Ascii
    - Convert from UTF-8 to ASCII
  - Iso8859_12Utf8
    - Convert from ISO 8859-1 to UTF-8
- Shows:
  - TEncoding (default and custom)
  - TStreamReader/TStreamWriter
  - THandleStream

Q & A

Jeroen Pluimers
better office benelux
jpluimers@better-office.com

If you have questions after the session, please mail me

Downloads will be on my blog: http://wiert.wordpress.com