

Re: how to communicate unsigned char* to Java

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.networks/2007-05/msg00236>

- *From:* Ali <abdulrazaq@xxxxxxxx>
 - *Date:* 25 May 2007 03:03:36 -0700
-

On May 24, 1:25 am, "Günter Prossliner" <g.prossliner@gmx.at> wrote:

Hello Ali!

Snip from GP:

This will not work, although it may work on some bunch of data.

And what is the data limit of this working solution? i mean the max size when this tech. will stop working.

It is not about size. It will stop working when a specific byte-sequences occur within the binary data which are no valid Unicode.

When you use ANSI Strings to read binary data

1. every byte stands for it's own
2. every possible value has an opposite Character within ANSI

==> you may read any binary data as an ANSI String. "String" is just an representation of the underlying binary data, so you can convert them without loosing information. Although in languages which supports an explicit binary type (in opposite to c++ which uses 'char' for character and binary), this was never a good coding practice.

As opposite to Unicode:

1. characters are encoded in more than one byte (when using UTF-16)
2. NOT every possible combination of values is an valid Unicode Code-Point.

What happens when an Unicode-Parser dedects an invalid byte-Sequence?

- * Thrown an error (in this case you will not be able to read the data at all)
- * Substitute the invalid sequence to something want may be SIMILAR (an

Re: how to communicate unsigned char* to Java

in this case you will not get the same binary data when you interpret this string a binary data).

* Skipp the invalid character (needless to say that you will loose data)

Just check out the UNICODE specification for details.

see:<http://www.unicode.org/reports/tr22/>(a document specifing an XML-format for UNICODE Exchange).

1.1 Illegal and Unassigned Codes

00 The sequence is illegal. There are two variants of this. First is where the sequence is incomplete. For example,

* 0xA3 is incomplete in CP950.

Unless followed by another byte of the right form, it is illegal.

* 0xD800 is incomplete in Unicode.

Unless followed by another value of the right form, it is illegal.

* 0xDC00 is incomplete in Unicode.

Unless preceded by another value of the right form, it is illegal.

The second variant is where the sequence is complete, but explicitly illegal. For example,

0xFFFF is illegal in Unicode. This value can never occur in valid Unicode text, and will never be assigned.

00 The source sequence represents a valid code point, but is unassigned (aka undefined). This sequence may be given an assignment in some future (evolved) version of the character encoding

.

For example,

* 0xA3 0xBF is unassigned in CP950, as of 1999.

* 0x0EDE is unassigned in Unicode, V3.0

00 The source sequence is assigned, but unmappable: there is no corresponding code point in the target encoding to accurately represent the source sequence.

For example,

The long dash is assigned in Unicode, but cannot be mapped to ISO-8859-1.

In the case of illegal source sequences, a conversion routine will typically provide the following options:

* stop (or throw an exception)

in particular, stopping is commonly used by higher level character encoding schemes, such as ISO 2022 conversions, to know when to stop converting into one encoding and pick another to convert to.

Re: how to communicate unsigned char* to Java

in either case, the information as to length of the bad sequence should be available and the conversion should be resumable (after the caller handles the bad sequence).

* skip the source sequence

while this is commonly an option, it can also hide corruption problems in the source text.

* map to a substitution character

such as the Unicode U+FFFD REPLACEMENT CHARACTER.

=====
see:http://unicode.org/faq/utf_bom.html#40
=====

Q: Are there any 16-bit values that are invalid?

A: The two values FFFE16 and FFFF16 as well as the 32 values from FDD016 to FDEF16 represent noncharacters. They are invalid in interchange, but may be freely used internal to an implementation. Unpaired surrogates are invalid as well, i.e. any value in the range D80016 to DBFF16 not followed by a value in the range DC0016 to DFFF16, or any value in the range DC0016 to DFFF16 not preceded by a value in the range D80016 to DBFF16. [AF]

=====
GP

GP! thanks for value thoughts and i enjoyed reading that. Well, i guess it seems another discussion like OO verses structural programming or good programming verses bad one;-)

ali

.