Character Code

Chen Kaiwang
kaiwang.chen@gmail.com

February 22, 2012

Sloppiness brings confusion.

Basic Concepts

- What's a Character
- Unicode Encoding Model
- GBK

Case Studies

- Webpage Browsing
- Directory Listing in Terminal

String Literals

- C, Java, Perl, PHP, JavaScript, MySQL
- Locale, i18n and L10n

Case Studies

- Accessing External Data
- The Myth Of MySQL Character Set Support
- Type In A Character
- Custom Encoding Scheme?

What's a character?

- a piece of information
- represented as a natural number, a.k.a. code point

  27721
- carried in octets

UTF-8	e6 b1 89
UTF-16BE	6c 49
- you see the glyph

Unicode

A character set standard which plans to codify all of the writing systems of the world, plus many other symbols.

Logical character v.s. actual character(codepoint)

A Unicdoe logical character can actually consist of more than one internal actual charcter or code point.

Unicode encoding model

Rather than mapping characters directly to octets (bytes), it separately define what characters are available, their numbering, how those numbers are encoded as a series of "code units" (limited-size numbers), and finally how those units are encoded as a stream of octets.

Unicode encoding model, cont

- character repertoire, C_r
- character set,

$$f(c) = n, \quad c \in C_r, \quad n \in \text{natural number}$$

- character encoding form(sequence of units)

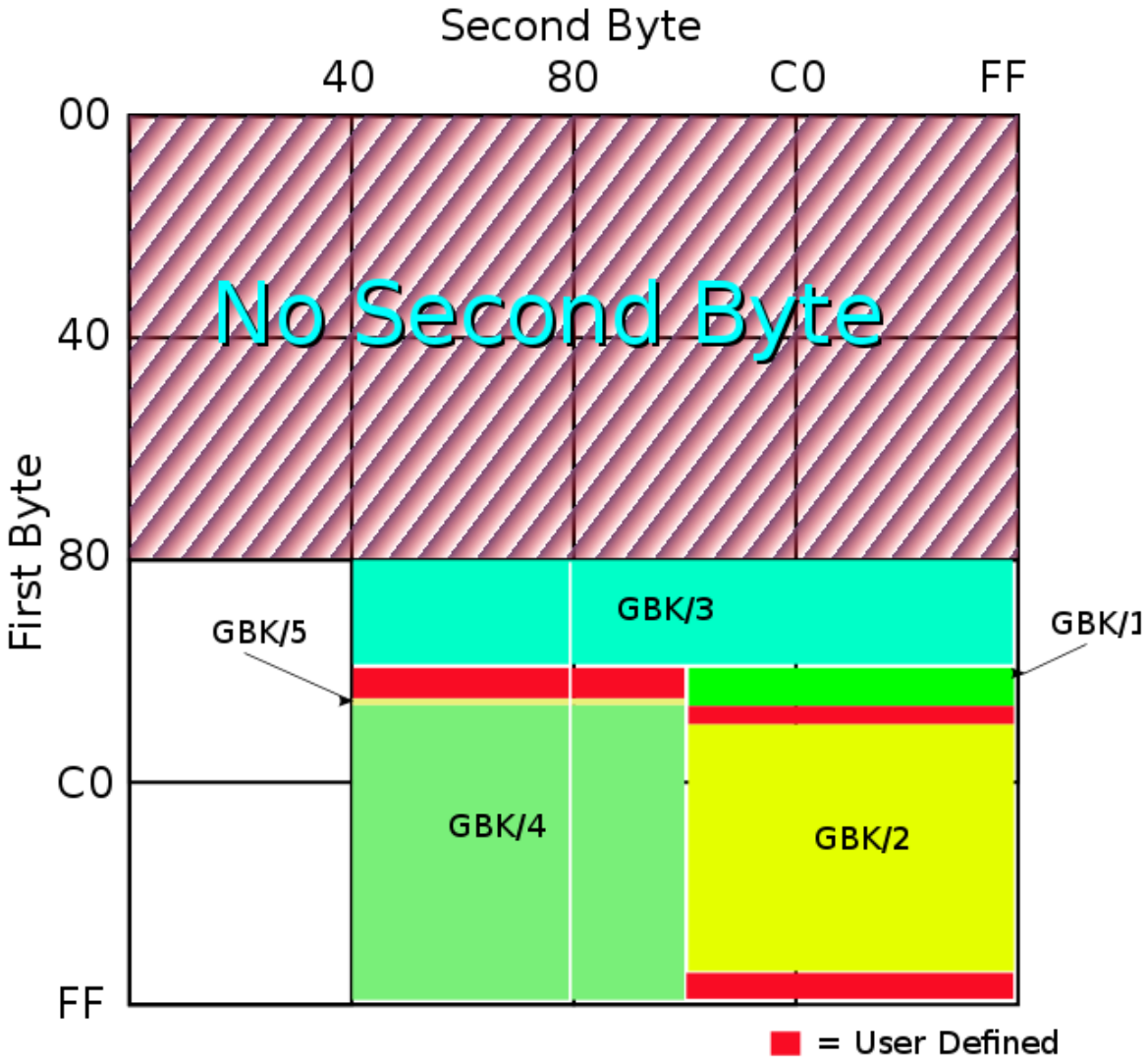
$$n = u_0 u_1 ..., \quad 0 \leq u_i < 2^{8N}, \quad i \geq 0$$

- character encoding scheme, a.k.a. endianess when using bigger-than-octet units

$$u_i = o_0 o_1 ... o_M = \begin{cases} b_{M,7} b_{M,6} ... b_{M,0} ..... b_{0,7} b_{0,6} ... b_{0,0} & \text{if little endian} \\ b_{0,7} b_{0,6} ... b_{0,0} ..... b_{M,7} b_{M,6} ... b_{M,0} & \text{if big endian} \end{cases}$$

- high level protocol, e.g.

```
<p xml:lang="en">The quick brown fox jumps over the lazy dog.</p>
```

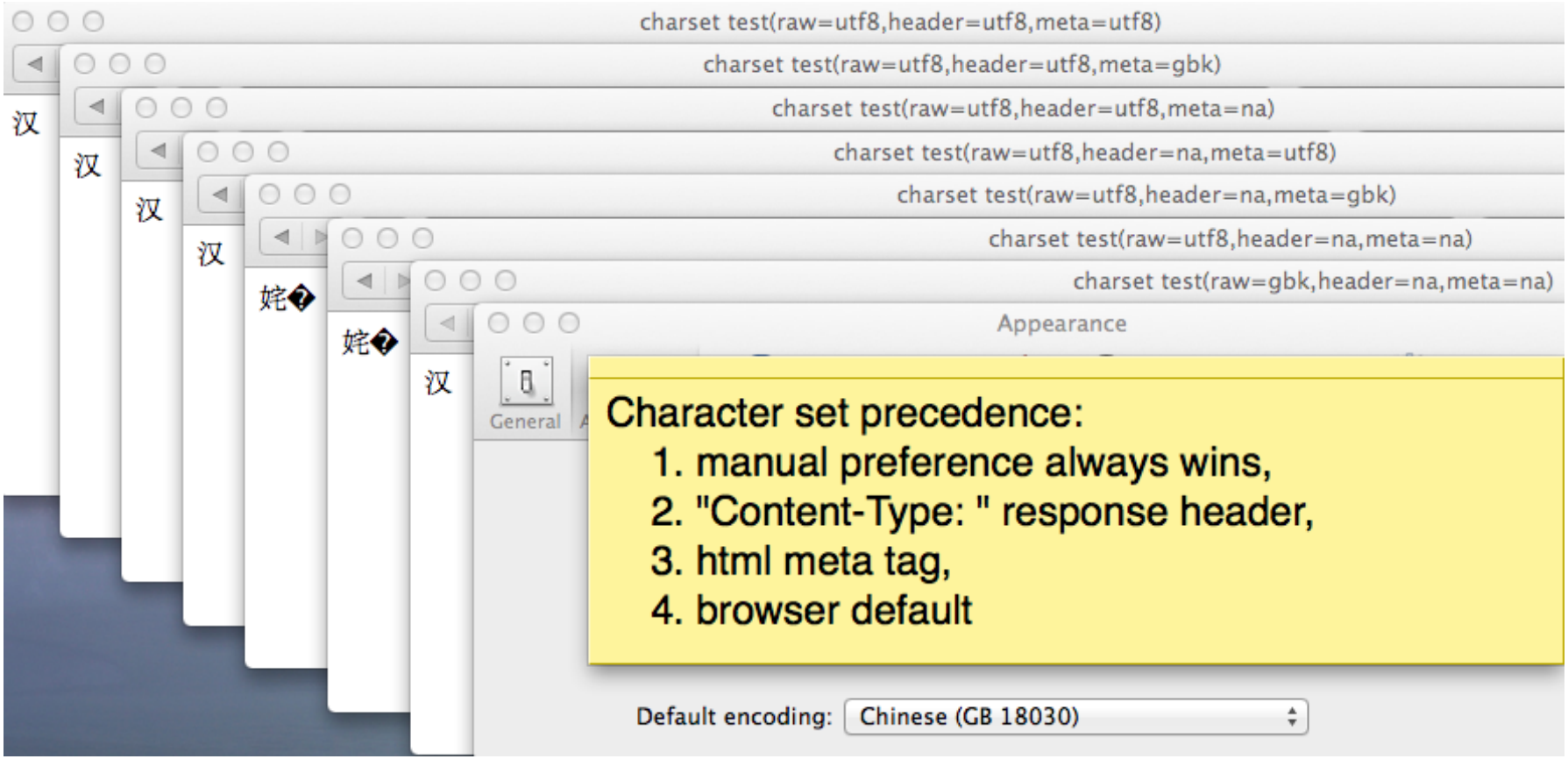


GBK

A character is encoded as 1 or 2 bytes.

A byte in the range 00-7F is a single byte that means the same thing as it does in ASCII.

Appearance



ps: browser may guess encoding between step 3 and 4.

The code

```
<?php
$raw = $_GET["raw"]; $hdr = $_GET["header"]; $text= $_GET["text"];
$hint = $_GET["hint"];
if ((strcmp($raw,"utf8") && strcmp($raw,"gbk")) ||
    (strcmp($hdr,"utf8") && strcmp($hdr,"gbk") && strcmp($hdr,"na")) ||
    (strcmp($text,"plain") && strcmp($text, "html")) ||
    (strcmp($hint,"utf8") && strcmp($hint,"gbk") && strcmp($hint,"na"))) {
  header("HTTP/1.0 403 Forbidden"); exit;
}
$b["utf8"]="\xe6\xb1\x89";
$b["gbk"] ="\xba\xba";
$h["utf8"]="Content-Type: text/".$text."; charset=utf8";
$h["gbk"] ="Content-Type: text/".$text."; charset=gbk";
$h["na"]  ="Content-Type: text/".$text;
$m["utf8"]='<meta http-equiv="Content-Type" content="text/html;charset=utf8">';
$m["gbk"] ='<meta http-equiv="Content-Type" content="text/html;charset=gbk">';
$m["na"]  ="";
header($h[$hdr]);
echo "<html><head><title>charset test(raw=$raw,header=$hdr,meta=$hint)</title>"
  .$m[$hint]."</head><body>".$b[$raw]."</body></html>";
```

Appearance

Terminal emulators

- GNOME Terminal
- Konsole
- SecureCRT
- PuTTY
- WinSCP, not a terminal but works similiarly

Internals

```
bash-3.2$ ls -lia
total 28
61251761 drwxr-xr-x  2 root root   73 Dec 28 02:34 .
75498088 drwxr-x--- 24 root root 8192 Dec 28 02:34 ..
60832834 -rwxr-xr-x  1 root root 7844 Dec 28 02:34 a.out
60832837 -rw-r--r--  1 root root  885 Dec 28 02:34 dirlisting.c
60832833 -rw-r--r--  1 root root    0 Dec 28 01:14 ??_gbk.txt
60832832 -rw-r--r--  1 root root    0 Dec 28 01:12 汉_utf8.txt
bash-3.2$ ./a.out .
ino=61251761, off=4, len=24, name=., bytes:
  0x2e,
ino=75498088, off=12, len=24, name=.., bytes:
  0x2e,0x2e,
ino=60832832, off=15, len=32, name=汉_utf8.txt, bytes:
  0xe6,0xb1,0x89,0x5f,0x75,0x74,0x66,0x38,0x2e,0x74,0x78,0x74,
ino=60832833, off=172, len=32, name=??_gbk.txt, bytes:
  0xba,0xba,0x5f,0x67,0x62,0x6b,0x2e,0x74,0x78,0x74,
ino=60832837, off=175, len=32, name=dirlisting.c, bytes:
  0x64,0x69,0x72,0x6c,0x69,0x73,0x74,0x69,0x6e,0x67,0x2e,0x63,
ino=60832834, off=512, len=32, name=a.out, bytes:
  0x61,0x2e,0x6f,0x75,0x74,
```

The code

```
/* many header includes omitted ... */
 12 int print_dent(struct dirent *dp) {
 13   printf("ino=%ld, off=%u, len=%d, name=%s, bytes:\n  ",
 14          dp->d_ino, dp->d_off, dp->d_reclen, dp->d_name);
 15   unsigned char *c;
 16   for (c=dp->d_name; *c; c++)
 17     printf("%#02x,",*c);
 18   printf("\n");
 19 }
 20 int main(int argc, char *argv[]) {
 21   struct dirent de; int ret;
 22   if (argc<2) { printf("Usage: %s DIR\n", argv[0]); return 1; }
 23   int fd = open(argv[1], O_RDONLY);
 24   if (fd < 0) { perror(""); return 1; }
 25   while ((ret= syscall(SYS_getdents,fd, &de, sizeof(de))) > 0) {
 26     print_dent(&de);
 27     lseek(fd,de.d_off,SEEK_SET);
 28   }
 29   if (ret == -1) { perror(""); return 1; }
 30 }
```

Literals

the values we write in a conventional form whose value is obvious, a.k.a. constants, explicit constants, manifest constants.

String literals

the representation of string values within the source code of a computer program.

- Notation
  - declarative notation
  - whitespace delimiters (indentation)
  - bracketed delimiters (quoting)
    delimiter collision
- Interpretation
  - raw string
  - metacharacters support
    variable interpolation, escape sequence

C: bytes in read-only section

Wide characters and long strings must use the prefix `L` when defined in quotes. Notice the `_TEXT` macro in Visual C++ maps to `L` if `_UNICODE` is defined.

Conversion is done by locale-sensitive `mbstowcs(3)` defined in C89, or locale-independent `iconv(3)` defined in X/Open Portability Guide, version 2.

Appearance of locale-sensitive conversion

```
-bash-3.2$ gcc -g -O0 literal.c
-bash-3.2$ LANG=zh_CN.UTF-8 ./a.out
literal vs. \x6c49: equal
0,27721,0x6c49
-bash-3.2$ LANG=zh_CN.GBK ./a.out
literal vs. \x6c49: not equal
0,23033,0x59f9
```

The code

```
/* many headers omitted ... */
  8 int main(int argc, char *argv[])
  9 {
 10   int i;
 11   wchar_t buf[10] = {0,};
 12   wchar_t *ws=L""; /* Han in utf-8 */
 13   char *s="";      /* Han in utf-8 */
 14
 15   setlocale(LC_CTYPE,"");
 16   mbstowcs(buf,s,sizeof buf / sizeof(wchar_t));
 17   buf[9] = 0;
 18
 19   printf("literal vs. \\x6c49: %s\n",
 20     (!wcscmp(buf,ws) ? "equal" : "not equal"));
 21   for (i=0;buf[i];i++)
 22     printf("%d,%d,%#0x\n",i,buf[i],buf[i]);
 23   return 0;
 24 }
```

Internals in AT&T Assembly

```
/* ... */
10          .section        .rodata
11          .align 4
12  .LC0:
13          .string "Il" /* 49 6c */
14          .string ""
15          .string ""
16          .string ""
17          .string ""
18          .string ""
19  .LC1:               /* e6 b1 89 */
20          .string "\346\261\211"
21  .LC2:
22          .string ""
23  .LC3:
24          .string "equal"
/* ... */
31          .text
32  .globl main
33          .type   main, @function
34  main:
35  .LFB5:
36          .file 1 "literal.c"
/* ... */
52          .loc 1 12 0
53          movq    $.LC0, -16(%rbp)
54          .loc 1 13 0
55          movq    $.LC1, -8(%rbp)
56          .loc 1 15 0
57          movl    $.LC2, %esi
58          movl    $0, %edi
59          call    setlocale
60          .loc 1 16 0
61          movq    -8(%rbp), %rsi
62          leaq    -64(%rbp), %rdi
63          movl    $10, %edx
64          call    mbstowcs
/* ... */
```

Java: CONSTANT_Utf8_info in constant_pool

```
-bash-3.2$ LANG=zh_CN.GBK javac literal_gbk.java
-bash-3.2$ LANG=en_US.UTF-8 java literal_gbk
literal vs. \u6c49: == equal
literal(1):  e6 b1 89
\u6c49(1):  e6 b1 89
-bash-3.2$ LANG=zh_CN.GBK java literal_gbk
literal vs. \u6c49: == equal
literal(1):  ba ba
\u6c49(1):  ba ba
-bash-3.2$ hexdump -C literal_gbk.class |grep -C 3 'e6 b1 89'
00000100  76 61 2f 6c 61 6e 67 2f  53 74 72 69 6e 67 3b 29  |va/lang/String;)|
00000110  56 07 00 50 01 00 0a 53  6f 75 72 63 65 46 69 6c  |V..P...SourceFil|
00000120  65 01 00 10 6c 69 74 65  72 61 6c 5f 67 62 6b 2e  |e...literal_gbk.|
00000130  6a 61 76 61 0c 00 1c 00  1d 01 00 03 e6 b1 89 07  |java............|
00000140  00 51 0c 00 52 00 53 01  00 17 6a 61 76 61 2f 6c  |.Q..R.S...java/l|
00000150  61 6e 67 2f 53 74 72 69  6e 67 42 75 69 6c 64 65  |ang/StringBuilde|
00000160  72 01 00 14 6c 69 74 65  72 61 6c 20 76 73 2e 20  |r...literal vs. |
-bash-3.2$ hexdump -C literal_gbk.java | grep -C 3 'ba ba'
00000050  20 76 6f 69 64 20 6d 61  69 6e 28 53 74 72 69 6e  | void main(Strin|
00000060  67 5b 5d 20 61 72 67 76  29 20 74 68 72 6f 77 73  |g[] argv) throws|
00000070  20 45 78 63 65 70 74 69  6f 6e 20 7b 0a 20 20 20  | Exception {.   |
00000080  20 53 74 72 69 6e 67 20  73 31 3d 20 22 ba ba 22  | String s1= ".."|
00000090  3b 0a 20 20 20 20 53 74  72 69 6e 67 20 73 32 3d  |;.    String s2=|
000000a0  22 5c 75 36 63 34 39 22  3b 0a 20 20 20 20 53 79  |"\u6c49";.    Sy|
000000b0  73 74 65 6d 2e 6f 75 74  2e 70 72 69 6e 74 6c 6e  |stem.out.println|
```

The code

```
 1 import java.io.*;
 2 import java.util.*;
 3 public class literal_gbk {
 4   public static void main(String[] argv) throws Exception {
 5     String s1= ""; // character Han in gbk
 6     String s2="\u6c49";
 7     System.out.println("literal vs. \\u6c49: " + (s1==s2 ? "==":"!=")
 8       + " " + (s1.equals(s2) ? "equal":"not equal"));
 9     System.out.print("literal("+s1.length()+"): "); print_bytes(s1);
10     System.out.print("\\u6c49("+s2.length()+"): "); print_bytes(s2);
11   }
12   public static void print_bytes(String s) {
13     byte[] b = s.getBytes();
14     for (int i=0; i<b.length; i++) {
15       System.out.print(" " + Integer.toHexString(0xFF & b[i]));
16     }
17     System.out.println();
18   }
19 }
```

Perl: SvUTF8 since 5.8

```
....We now view strings not as sequences of bytes, but as sequences
of numbers in the range 0 .. 2**32-1 (or in the case of 64-bit
computers, 0 .. 2**64-1) -- Programming Perl, 3rd ed.
```

the two fundamentally different kinds of strings and string-operations in Perl:

- byte-oriented mode when the internal UTF8 flag is off
- character-oriented mode when the internal UTF8 flag is on

Perl code

```
 1 #!/usr/bin/perl -w
 2 use strict;
 3 use Devel::Peek;
 4 { no encoding;
 5   # Han in gbk
 6   my $x = '';
 7   Dump($x);
 8 }
 9 { use encoding "gbk";
10   # Han in gbk
11   my $x = '';
12   Dump($x);
13 }
```

Appearance

Notice \272\272 is hexadecimal baba which is Han in gbk.

```
-bash-3.2$ perl encoding_gbk.pl
SV = PV(0x1ae8160) at 0x1ae7f90
  REFCNT = 1
  FLAGS = (PADBUSY,PADMY,POK,pPOK)
  PV = 0x1aff4e0 "\272\272"\0
  CUR = 2
  LEN = 8
SV = PV(0x1ae81d8) at 0x1ae7f80
  REFCNT = 1
  FLAGS = (PADBUSY,PADMY,POK,pPOK,UTF8)
  PV = 0x1afbe30 "\346\261\211"\0 [UTF8 "\x{6c49}"]
  CUR = 3
  LEN = 8
```

PHP

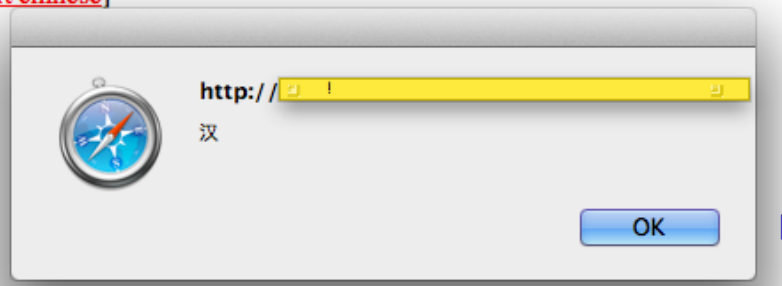
The string in PHP is implemented as an array of bytes and an integer indicating the length of the buffer. String literals will be encoded in whatever fashion it is encoded in the script file, or if Zend Multibyte is enabled, the script may be written in an arbitrary encoding and then converted to a certain internal encoding.

```
php-5.3.6/configure
105666 if test "${enable_zend_multibyte+set}" = set; then
105667   enableval="$enable_zend_multibyte"
105668
105669   ZEND_MULTIBYTE=$enableval
105670
105671 else
105672
105673   ZEND_MULTIBYTE=no
105674
105675 fi

-bash-3.2$ php-config --configure-options | tr ' ' '\n' | grep zend
-bash-3.2$
```

Encoding in JavaScript

PHP works, cs=gbk, js=gbk.js [alert chinese]




```
--- gbk.js ---
var s = '汉';
function alert_chinese() {
  alert(s);
}

--- php ---
<?php
$cs="utf8";
$js="utf8.js";
if (isset($_GET["cs"])&&preg_match('/^(?:utf8|gbk)$/',$_GET["cs"]))
  $cs=$_GET["cs"];
if (isset($_GET["js"])&&preg_match('/^(?:utf8|gbk)[.]js$/',$_GET["js"]))
  $js=$_GET["js"];

header("Cache-Control: max-age=0");
echo "PHP works, cs=$cs, js=$js";
echo "
<html>
<head>
<script type='text/javascript' charset=$cs src=$js>
</script>
</head>
<body>
[<a href='javascript:alert_chinese()'>alert chinese</a>]
<pre>
--- $js ---
".htmlspecialchars(file_get_contents(dirname($_SERVER["SCRIPT_FILENAME"])."/".$js))."
--- php ---
".htmlspecialchars(file_get_contents($_SERVER["SCRIPT_FILENAME"]))."
</body>
</html>
";
```

- standalone script, declared by HTML `script charset=gbk`
- embedded script, same to that of page
- interpreter accepts UTF-16 source text
- JSON text uses Unicode, encoding guessed from initial two octets
- XMLHttpRequest MIME declaration

MySQL: every string literal has a character set and a collation

`SELECT 'string'`, defined by `character_set_connection` and `collation_connection` system variables.

`_x` introducer, indicates the character set `x` for the following string, but does not change how the parser performs escape processing within the string. Escapes are always interpreted by the parser according to the character set given by `character_set_connection`.

Internationalization(i18n)

the process of designing a software application so that it can be adapted to various languages and regions without engineering changes

Localization(L10n)

the process of adapting internationalized software for a specific region or language by adding locale-specific components and translating text.

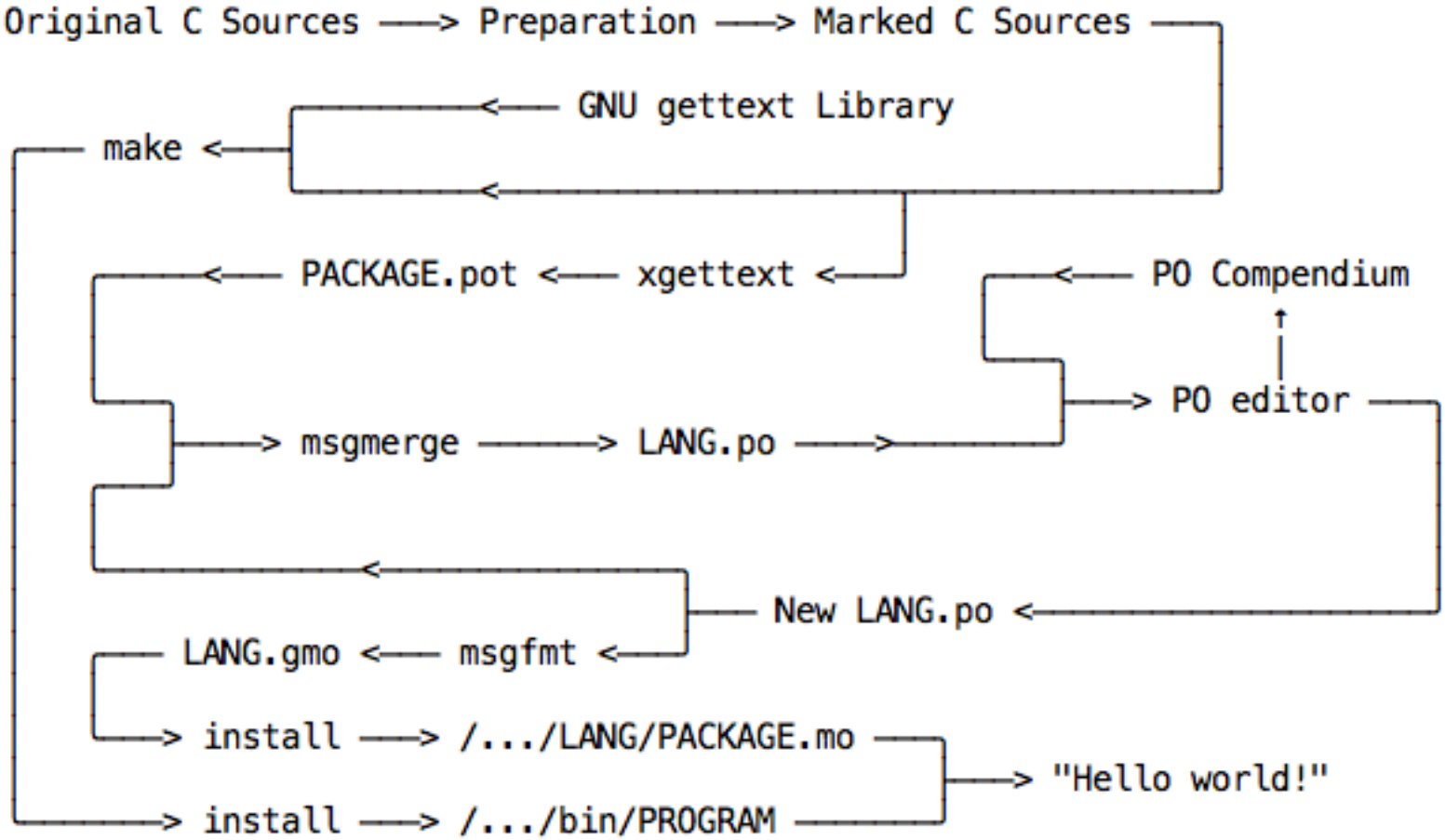
Locale

a set of parameters that defines the user's language, country and any special variant preferences that the user wants to see in their user interface , including UI language, input, display (time/date, number and currency), etc. Locale identifier:
`[language[_territory][.codeset][@modifier]]`

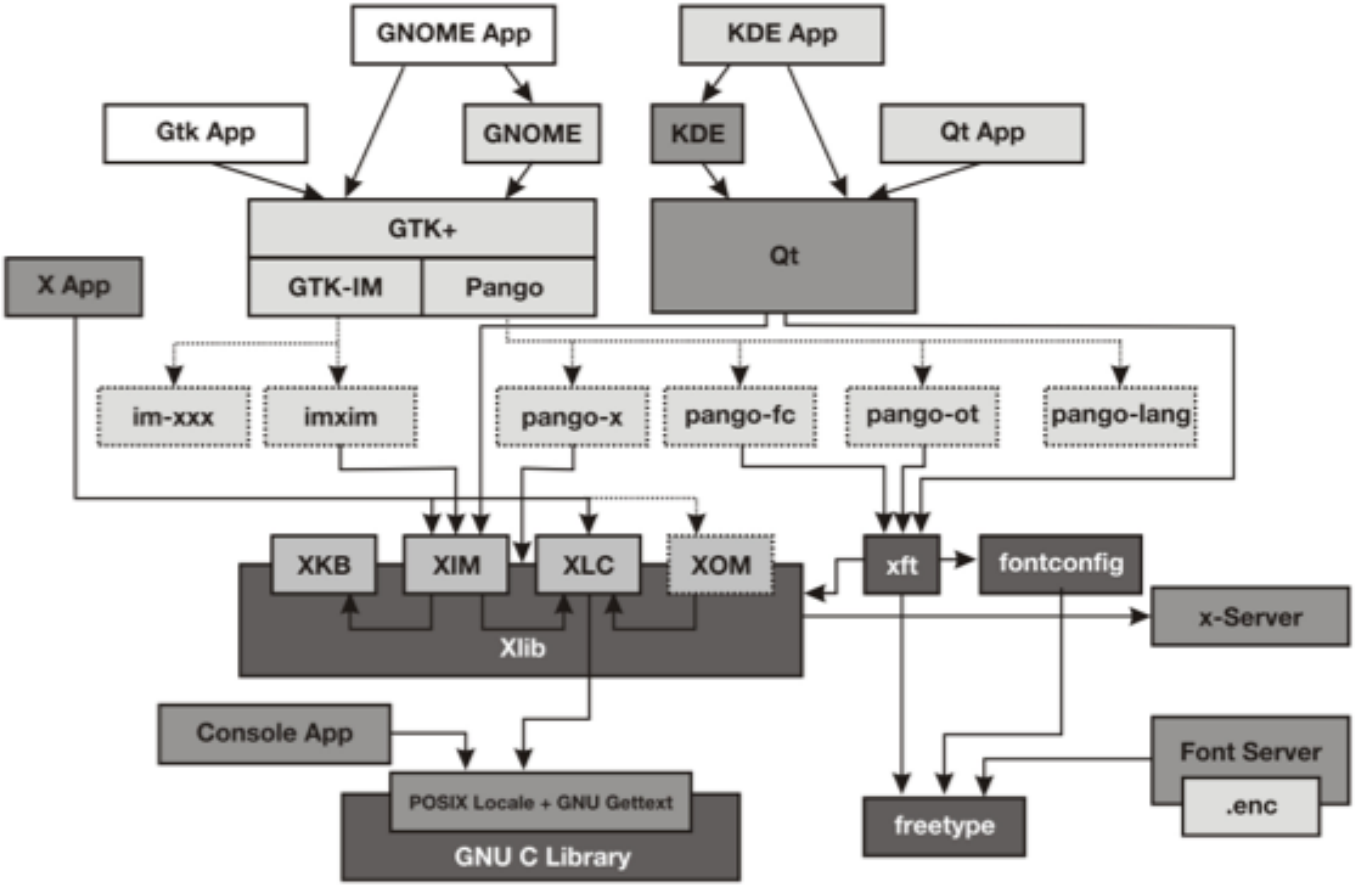
To internationalize a UI, every text string employed in interaction, a.k.a. resource string, must be translated into all supported languages; then all output of literal strings, and literal parsing of input in UI code must be replaced by hooks to i18n libraries.

- GTK+/GNOME: GNU gettext interface
- KDE/Qt: tr()

GNU gettext working process



GNU/Linux Desktop Structure and Internationalization



The two types of streams

In the Unix world, stream of bytes model is used for file I/O, socket I/O, and terminal I/O.

Comparison Of Streaming Access

Language	Stream of Bytes	Character Streams
C	read(2)/write(2), stdio char *b;	multibyte or wide characters unsigned char *s; wchar_t *s;
Java	InputStream/OutputStream byte[] b;	Reader/Writer String s; char[] s;
Perl	:unix:perlio scalar with SvUTF8 off	:unix:perlio:encoding(gbk) scalar with SvUTF8 on/off(8-bit)

The two types of streams, continued

Conversion Between Bytes And Characters

Language	Chareacters To Bytes	Bytes To Characters
C	typecast(8 bit), wcstombs(3)/mbstowcs(3), iconv(3)	
Java	str.getBytes(Charset cs) Writer(OuputStream os)	String(byte[] b, Charset cs) Reader(InputStream is)
Perl	Encode::encode(), unpack	Encoode::decode(), pack
	specify PerlIO layers: binmode, open, open pragma, perl -CSA	

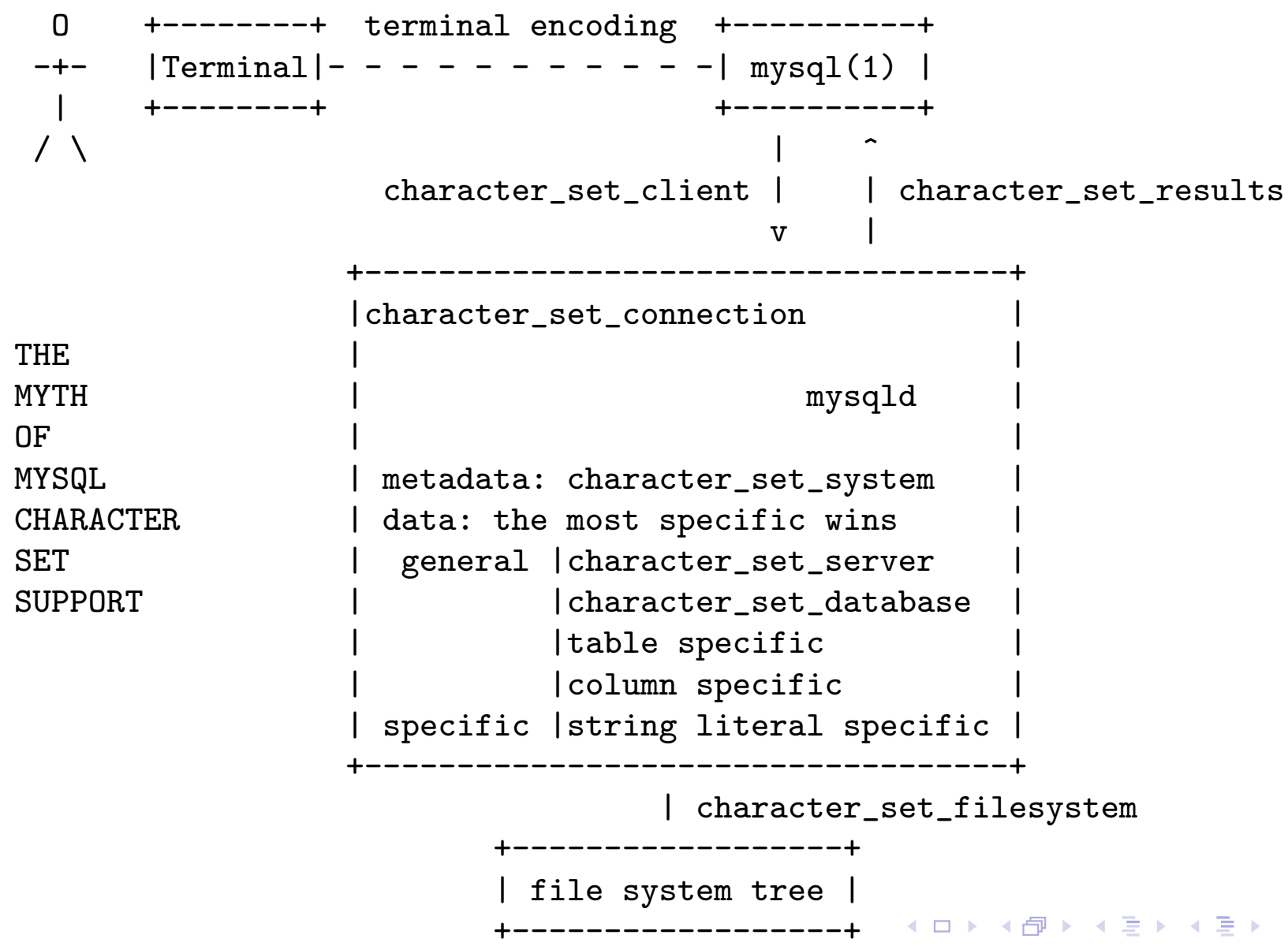
SvUTF8 flag: the literal, file and DBD::mysql problem

```
 1 #!/usr/bin/perl -w
 2 use strict; use Encode; use DBI qw(:utils); use Devel::Peek;
 3 (my $dbh = DBI->connect("dbi:mysql:test:mysql_socket=...") or die "conn $!")->do("set names utf8");
 4 open FH, "<", "han_gbk.txt" or die "open $!"; chomp(my $bytes = <FH>); close FH;
 5 open FH, "<:encoding(gbk)", "han_gbk.txt" or die "gbk $!"; chomp(my $chars = <FH>); close FH;
 6 my $db_bytes = $dbh->selectall_arrayref("SELECT utf8 FROM han")->[0][0];
 7 my $db_chars = decode("utf-8", $db_bytes);
 8 print "==== bytes: ".data_string_desc($bytes) . "\n"; Dump($bytes);
 9 print "==== chars: ".data_string_desc($chars) . "\n"; Dump($chars);
10 print "==== db bytes: ".data_string_desc($db_bytes) . "\n"; Dump($db_bytes);
11 print "==== db chars: ".data_string_desc($db_chars) . "\n"; Dump($db_chars);
12 {  my $re = qr/()/; print "==== regexp default encoding: \n"; # Han in utf-8
13 print "bytes: got ", ($bytes =~ $re ? $1 : "N/A"), "\n";
14 print "chars: got ", ($chars =~ $re ? $1 : "N/A"), "\n";
15 print "db bytes: got ", ($db_bytes =~ $re ? $1 : "N/A"), "\n";
16 print "db chars: got ", ($db_chars =~ $re ? $1 : "N/A"), "\n";   }
17 {  use utf8; my $re = qr/()/; print "==== regexp utf8: \n";   # Han in utf-8
18 print "bytes: got ", ($bytes =~ $re ? $1 : "N/A"), "\n";
19 my $x = "chars: got ". ($chars =~ $re ? $1 : "N/A"). "\n"; print $x;
20 print "db bytes: got ", ($db_bytes =~ $re ? $1 : "N/A"), "\n";
21 print "db chars: got ", ($db_chars =~ $re ? $1 : "N/A"), "\n";   }
```

```
-bash-3.2$ perl SvUTF8.pl
==== bytes: UTF8 off, non-ASCII, 2 characters 2 bytes
SV = PV(0x96ab80) at 0x942e00
  REFCNT = 1
  FLAGS = (PADBUSY,PADMY,POK,pPOK)
  PV = 0xb717b0 "\272\272"\0
  CUR = 2
  LEN = 80
==== chars: UTF8 on, non-ASCII, 1 characters 3 bytes
SV = PV(0xafc198) at 0x9ad3e0
  REFCNT = 1
  FLAGS = (PADBUSY,PADMY,POK,pPOK,UTF8)
  PV = 0x8deb50 "\346\261\211"\0 [UTF8 "\x{6c49}"]
  CUR = 3
  LEN = 80
==== db bytes: UTF8 off, non-ASCII, 3 characters 3 bytes
SV = PV(0xafc1f8) at 0x9ad3a0
  REFCNT = 1
  FLAGS = (PADBUSY,PADMY,POK,pPOK)
  PV = 0xb7dab0 "\346\261\211"\0
  CUR = 3
  LEN = 8
==== db chars: UTF8 on, non-ASCII, 1 characters 3 bytes
SV = PV(0xafc708) at 0x9ad310
  REFCNT = 1
  FLAGS = (PADBUSY,PADMY,POK,pPOK,UTF8)
  PV = 0xb74de0 "\346\261\211"\0 [UTF8 "\x{6c49}"]
  CUR = 3
  LEN = 8
==== regexp default encoding:
bytes: got N/A
chars: got N/A
db bytes: got 汉
db chars: got N/A
==== regexp utf8:
bytes: got N/A
Wide character in print at SvUTF8.pl line 19.
chars: got 汉
db bytes: got N/A
Wide character in print at SvUTF8.pl line 21.
db chars: got 汉
```

What's the problem

- DBD::mysql assumes stream of bytes, unless 4.004 and above with `mysql_enable_utf8` on
- Windows assumes cp936 (gbk) encoding; Linux is aware of locale, and UTF-8 is common case. Accessing text files, including source code, suffer from inconsistency across OS'es
- Perl: regular expressions match against characters; standard streams default to 8-bit



THE MYTH OF MYSQL CHARACTER SET SUPPORT

```
# mysql --no-defaults
mysql> CREATE TABLE test.session_variables_nodefault
  SELECT * FROM information_schema.session_variables
  WHERE variable_name LIKE 'character_set\_%';
mysql> SET NAMES utf8;
mysql> CREATE TABLE test.session_variables_setutf8
  SELECT * FROM information_schema.session_variables
  WHERE variable_name LIKE 'character_set\_%';
mysql> SELECT t1.variable_name AS name,
  t1.variable_value AS value_nodefault, t2.variable_value AS value_utf8
  FROM test.session_ variables_nodefault t1 LEFT JOIN
   test.session_variables_setutf8 t2 ON t1.variable_name = t2.variable_name;
+--------------------------+-----------------+------------+
| name                     | value_nodefault | value_utf8 |
+--------------------------+-----------------+------------+
| CHARACTER_SET_CONNECTION | latin1          | utf8       |
| CHARACTER_SET_RESULTS    | latin1          | utf8       |
| CHARACTER_SET_SERVER     | utf8            | utf8       |
| CHARACTER_SET_FILESYSTEM | binary          | binary     |
| CHARACTER_SET_DATABASE   | utf8            | utf8       |
| CHARACTER_SET_SYSTEM     | utf8            | utf8       |
| CHARACTER_SET_CLIENT     | latin1          | utf8       |
+--------------------------+-----------------+------------+
```

```
mysql> show create database test\G
*************************** 1. row ***************************
       Database: test
Create Database: CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET utf8 */

mysql> show create table test.han\G
*************************** 1. row ***************************
       Table: han
Create Table: CREATE TABLE `han` (
  `utf8` varchar(10) NOT NULL,
  `gbk` varchar(20) CHARACTER SET gbk NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8

mysql> select length(utf8), length(gbk) from test.han;
+--------------+-------------+
| length(utf8) | length(gbk) |
+--------------+-------------+
|            3 |           2 |
+--------------+-------------+
```

Type In A Character

- rawscan code, represent the key position
- keycode, translated by keyboard model
- keyboard symbol, a.k.a. keysym, translated by keyboard layout (qwerty, dvorak, etc)
- sequence of keysyms are translated to a character by the input method

Appearance

```
-bash-3.2$ java decode
-SM-http%3a%2f%2fwiki%2edev%2ewanmei%2ecom%2fdevmw%2findex%2ephp%2f-%E5-%A4-%A7-%E5-%AF-%8C-%E7-%BF-%81
http://wiki.dev.wanmei.com/devmw/index.php/大富翁
```

Custom Encoding Scheme?

- javascript: encodeURIComponent
- additionaly escape the special percent sign
- UTF-8 is backward-compatible with ASCII

```
 1 public class decode {
 2   private static String unencodeTarget(String paramString)
throws java.io.UnsupportedEncodingException {
 3     String str1 = "";
 4     int i = 0;
 5     String str2 = "$SM$";
 6     String str3 = "-SM-";
 7     int j = 4;
 8     String str4 = paramString;
 9     int k = (str4.startsWith(str3)) ? 45 : 36;
10     if ((paramString.startsWith(str2)) || (paramString.startsWith(str3))) { i = j; }
11     for (int l = i; l < paramString.length(); ++l) {
12       if (paramString.charAt(l) == k) {
13         if (paramString.charAt(l + 1) == k) {
14           l += 1;
15           str1 = str1 + paramString.charAt(l);
16         }
17       } else if (paramString.charAt(l) == '%') {
18         str1 = str1 + (char)Integer.parseInt(paramString.substring(l + 1, l + 3), 16);
19         l += 2;
20       } else { str1 = str1 + paramString.charAt(l); }
21     }
22     return new String(str1.getBytes("ISO-8859-1"),"UTF-8");
23   }
24   public static void main(String args[]) throws Exception {
24     String x =
"-SM-http%3a%2f%2fwiki%2edev%2ewanmei%2ecom%2fdevmw%2findex%2ephp%2f-%E5-%A4-%A7-%E5-%AF-%8C-%E7-%BF-%81";
26     System.out.println(x);
27     System.out.println(unencodeTarget(x));
28   }
29 }
```

Checklist

Are they bytes or characters?

Is it literal or external data stream?

What's the contract of data exchange?

Is the program aware of i18n?

Thanks

References

[1] Charles Petzold. CODE The Hidden Language of Computer Hardware and Software, 2000, ISBN:0-7356-1131-9

[2] Character encoding http://en.wikipedia.org/wiki/Character_set

[3] perldoc: perluniintro, perlunicode, Encode, PerlIO

[4] Unihan Database Lookup http://unicode.org/charts/unihan.html

[5] GBK, Chinese Internal Code Specification http://en.wikipedia.org/wiki/GBK

[6] String Literal http://en.wikipedia.org/wiki/String_literal

[7] Dennie Van Tassel. Literals http://www.gavilan.edu/csis/languages/literals.html

[8] Data Type Mappings http://msdn.microsoft.com/en-us/library/se784sk6.aspx

[9] PHP String Literal http://php.net/manual/en/language.types.string.php

[10] Character Literals and String Literals In Java
http://java.sun.com/docs/books/jls/second_edition/html/lexical.doc.html

[11] VM Spec The class File Format
http://java.sun.com/docs/books/jvms/second_edition/html/ClassFile.doc.html

[12] MySQL Character String Literal Character Set and Collation
http://dev.mysql.com/doc/refman/5.1/en/charset-literal.html

[13] GNU 'gettext' utilities http://www.gnu.org/software/gettext/manual/gettext.html

[14] Internationalization with Qt http://developer.qt.nokia.com/doc/qt-4.8/internationalization.html