Character Encodings and Unicode

This chapter describes how characters are encoded into bits.
After this chapter you should be able to convert between characters, Unicode code points and their UTF-8 encodings.

Unicode Character Encodings

Data often represents printable characters.

A standard called Unicode assigns a unique number, called a “code point,” to over 100,000 of the characters used by more than 100 languages and scripts. Unicode is used by most operating systems and Internet applications.

Exercise 1: How many bits would be required to uniquely identify 100,000 different characters? (Hint: $2^{16} = 65536$).

Each character could be represented with a 32-bit (4 byte) number. However, the UTF-8 (Unicode Transformation Format - 8 bits) format is widely used since it allows many documents to be stored or transmitted using fewer bytes. It also has some practical advantages[1]

ASCII (American Standard Code for Information Interchange) was an earlier character encoding that used 7 bits to encode letters from the English alphabet, numbers, and the most common punctuation symbols. UTF-8 is exactly the same as ASCII for these first 128 (2^7) characters. This means that ASCII documents are already encoded as UTF-8.

The table below shows the ASCII table which is the first “code chart” from the Unicode standard. The columns are labelled with the most significant (first) hex digit and the rows with the least-significant (second) hex digit of the numerical value of each character.

ASCII also includes some non-printable control codes (values 0 to 31) that were used to control printers. For example the line feed (LF) character would move the paper in the printer up one line.

Other Unicode characters require between 2 and 4 bytes according to the rules summarized in Table 3-6 of the Unicode standard shown below. Unicode values between 128 and 2047 include most characters from European languages and can be encoded in two bytes. Values from 2048 to 65535 include most CJK (Chinese, Japanese and Korean) characters and require three bytes. Some rarely-used symbols (e.g. emoticons or Mahjong tiles) have four-byte encodings.

[1]The UTF-8 encoding has no zero bytes and all values <0x80 represent ASCII characters.

Encoding to UTF-8

Step 1 From the value of the character’s code point choose a sequence of bytes (all numbers in hexadecimal):

code point	prefixes
0 – 7F	00
80 – 07FF	C0 80
800 – FFFF	E0 80 80
> FFFF	F0 80 80 80

Step 2 Convert the code point to binary and divide the value into groups of 6 bits.

Step 3 Starting at the right, add each group of 6 bits to the corresponding byte, starting at the right.

Example

The codepoint for the CJK character for potato (藷) is U+85F7. From the table above, this must be encoded into the three bytes E0, 80 and 80. The code point in binary is 1000 0101 1111 0111. The groups of 6 bits, starting on the right are thus 11 0111 (37), 01 01 11 (17), and 00 1000 (08). Adding these to the prefixes the bytes in the UTF-8 encoding are (E0+08=E8), (80+17=97), and (80+37=B7).

Exercise 2: The Chinese character for “Rice” (the grain) is 米 with Unicode value (code point) U+7C73. What is the UTF-8 encoding for this character?

Decoding from UTF-8

The most significant nybble of each byte in a UTF-8 sequence identifies its purpose:

Table 3-6. UTF-8 Bit Distribution

Scalar Value	First Byte	Second Byte	Third Byte	Fourth Byte
00000000 0xxxxxxx	0xxxxxxx			
00000yyy yyxxxxxx	110yyyyy	10xxxxxx		
zzzzyyyy yyxxxxxx	1110zzzz	10yyyyyy	10xxxxxx	
000uuuuu zzzzyyyy yyxxxxxx	11110uuu	10uuzzzz	10yyyyyy	10xxxxxx

nybble	meaning
0 – 7	ASCII, one byte
8, 9, A, or B	the second, third or fourth byte of an encoding
C or D	start of a 2-byte encoding
E	start of a 3-byte encoding
F	start of a 4-byte encoding

Step 1 Scan for a byte whose most significant nybble is not 8, 9, A or B. This will be the first byte of a UTF-8 sequence.

Step 2 Extract the appropriate number of bytes (1 through 4).

Step 3 Delete the bits indicated by 0 or 1 in Table 3-6 above.

Step 4 Concatenate the remaining bits. This is the code point in binary. Group every 4 bits starting on the right and convert the nybbles to hex.

Example

Find the first Unicode character in the UTF sequence BC, D0, BE. First we ignore the byte BC because the initial nybble is B. Then we extract 2 bytes because the most significant nybble of the byte D0 indicates a 2-byte encoding. Then we convert the two bytes to binary: D0 = 1101 0000 and BE = 1011 1110. Deleting the leading three bits from the first byte and the leading two bits from the second we are left with 1 0000 11 1110. Grouping into nybbles this is 0100 0011 1110 and converting to hexadecimal this is 43F. Looking this up in the Unicode charts (or on unicode.org) show this is the character п, "CYRILLIC SMALL LETTER PE."

Exercise 3: Find the codepoint of the first Unicode character in the sequence of bytes: `A0 88 EB 8C 80 EC`.

Text versus Binary Number Representations

It's important to understand the difference between text that represents a number and binary data. For example, the character '**1**' would be transmitted with a UTF-8 encoding as the byte 0x31 while a byte with the value 1 could be transmitted as 0x01.

Numbers can be stored in files or transmitted over communication systems in either binary format (e.g. one 8-bit value per byte) or in text format (as a sequence of numeric characters). Numbers in text format can be more easily interpreted by humans since they are sequences of printable characters.

Exercise 4: Four numbers are transmitted as the following CSV file:

```
2, 1
9, 3
```

How many bytes are required to transmit these four numbers formatted this way? Note that a "line feed" character is required at the end of each line and that spaces and commas also need to be transmitted.

How many bytes are required to transmit these four numbers if they are transmitted, one after another, if each is encoded as a 16-bit number? What if each number was encoded as a 32-bit number?

<table>
<tr><th></th><th>000</th><th>001</th><th>002</th><th>003</th><th>004</th><th>005</th><th>006</th><th>007</th></tr>
<tr><td>0</td><td>NUL
0000</td><td>DLE
0010</td><td>SP
0020</td><td>0
0030</td><td>@
0040</td><td>P
0050</td><td>`
0060</td><td>p
0070</td></tr>
<tr><td>1</td><td>SOH
0001</td><td>DC1
0011</td><td>!
0021</td><td>1
0031</td><td>A
0041</td><td>Q
0051</td><td>a
0061</td><td>q
0071</td></tr>
<tr><td>2</td><td>STX
0002</td><td>DC2
0012</td><td>"
0022</td><td>2
0032</td><td>B
0042</td><td>R
0052</td><td>b
0062</td><td>r
0072</td></tr>
<tr><td>3</td><td>ETX
0003</td><td>DC3
0013</td><td>#
0023</td><td>3
0033</td><td>C
0043</td><td>S
0053</td><td>c
0063</td><td>s
0073</td></tr>
<tr><td>4</td><td>EOT
0004</td><td>DC4
0014</td><td>$
0024</td><td>4
0034</td><td>D
0044</td><td>T
0054</td><td>d
0064</td><td>t
0074</td></tr>
<tr><td>5</td><td>ENQ
0005</td><td>NAK
0015</td><td>%
0025</td><td>5
0035</td><td>E
0045</td><td>U
0055</td><td>e
0065</td><td>u
0075</td></tr>
<tr><td>6</td><td>ACK
0006</td><td>SYN
0016</td><td>&
0026</td><td>6
0036</td><td>F
0046</td><td>V
0056</td><td>f
0066</td><td>v
0076</td></tr>
<tr><td>7</td><td>BEL
0007</td><td>ETB
0017</td><td>'
0027</td><td>7
0037</td><td>G
0047</td><td>W
0057</td><td>g
0067</td><td>w
0077</td></tr>
<tr><td>8</td><td>BS
0008</td><td>CAN
0018</td><td>(
0028</td><td>8
0038</td><td>H
0048</td><td>X
0058</td><td>h
0068</td><td>x
0078</td></tr>
<tr><td>9</td><td>HT
0009</td><td>EM
0019</td><td>)
0029</td><td>9
0039</td><td>I
0049</td><td>Y
0059</td><td>i
0069</td><td>y
0079</td></tr>
<tr><td>A</td><td>LF
000A</td><td>SUB
001A</td><td>*
002A</td><td>:
003A</td><td>J
004A</td><td>Z
005A</td><td>j
006A</td><td>z
007A</td></tr>
<tr><td>B</td><td>VT
000B</td><td>ESC
001B</td><td>+
002B</td><td>;
003B</td><td>K
004B</td><td>[
005B</td><td>k
006B</td><td>{
007B</td></tr>
<tr><td>C</td><td>FF
000C</td><td>FS
001C</td><td>,
002C</td><td><
003C</td><td>L
004C</td><td>\
005C</td><td>l
006C</td><td>|
007C</td></tr>
<tr><td>D</td><td>CR
000D</td><td>GS
001D</td><td>-
002D</td><td>=
003D</td><td>M
004D</td><td>]
005D</td><td>m
006D</td><td>}
007D</td></tr>
<tr><td>E</td><td>SO
000E</td><td>RS
001E</td><td>.
002E</td><td>>
003E</td><td>N
004E</td><td>^
005E</td><td>n
006E</td><td>~
007E</td></tr>
<tr><td>F</td><td>SI
000F</td><td>US
001F</td><td>/
002F</td><td>?
003F</td><td>O
004F</td><td>_
005F</td><td>o
006F</td><td>DEL
007F</td></tr>
</table>