2 Events, characters, codes, glyphs

As computer users, we tend to identify

the keystrokes we enter,

the characters we obtain,

their representation in a file,

and their visual appearance on the screen.

In reality, things are a bit more complicated, though.

Characters vs. Glyphs

Characters are distinguished by their meaning.

Glyphs are distinguished by their visual appearance.

Several characters may share the same glyph:

e. g., Latin “H” and Greek “Eta”.

Conversely, characters may have more than one glyph:

e. g., different forms of Latin letter “a”.

even in a single font, e. g., in Arabic.

Or no glyph at all:

e. g., the LINEFEED character.

Characters Sets

A character repertoire is a finite set of characters.

A coded character set maps the character of a character repertoire uniquely to natural numbers.

Examples:

7 bit:

ASCII
“national variants” of ASCII

8 bit:

ISO-8859-1, -2, -3, ...
MS-Windows codepage 1252, ...
MS-DOS codepages
Macintosh codepages

ISO-8859-1

See additional file `isocommented.txt`.

Beyond 8 Bits

Two approaches to deal with the plethora of character sets:

ISO-2022

Use escape sequences to change between different character sets.

Unicode/ISO-10646

Map all characters into a single character set.

Unicode/ISO-10646

Originally planned as a 16 bit character set (65,536 characters).

Since Version 2.0: surrogate mechanism makes it possible to encode at most 1,112,064 characters.

The first 256 characters of Unicode agree with ISO-8859-1.

Still, if Unicode characters are encoded by two (or four) bytes, Unicode is incompatible with ASCII or ISO-8859-1.

Unicode: Character Encodings

UTF-32/UCS-4:

Every character is encoded using 32 bits.

UTF-16/UCS-2:

Every character with a number below 65,536 is encoded using 16 bits; characters with numbers above 65,536 are encoded using two surrogate characters (16 + 16 bits).

UTF-8:

Every character is encoded using one to four bytes:

Unicode range U+0000 to U+007F: 1 byte.
00000000 00000000 0zzzzzzz → 0zzzzzzz

Unicode range U+0080 to U+07FF: 2 bytes.
00000000 00000yyy yyzzzzzz → 110yyyyy 10zzzzzz

Unicode range U+0800 to U+FFFF: 3 bytes.
00000000 xxxxyyyy yyzzzzzz → 1110xxxx 10yyyyyy 10zzzzzz

Unicode range U+010000 to U+1FFFFF: 4 bytes.
...

Features of UTF-8:

ASCII-compatible, hence Unix kernel and file system can deal with UTF-8 data.

First byte of a multibyte sequence is always recognizable, hence byte-oriented string search algorithms can be used with UTF-8 data.

Unicode

Don't expect a perfect solution:

Round-trip compatibility.
Design by committee.
Restrictions of existing hard- and software.
Human factors.

Consequences:

Glyphs as characters.
Normalized vs. non-normalized characters.
Language dependencies.
...

Useful Tools

recode, iconv:

Convert between different character sets/encodings.

expand:

Convert tabs to spaces

Locale environment variables `LC_CTYPE`, `LC_COLLATE`, ...:

Change the behaviour of the character handling, classification, and comparison functions.

Characters and Events

Modern keyboards send events, rather than characters to an application.

Not every event corresponds to a character:

`<Shift>`, `<Ctrl><Esc>`

Some processing may take place:

dead keys, input methods.

Beware: An application running in a terminal emulator such as xterm receives characters rather than events.

The conversion is done by the terminal emulator.

Useful Tools

xmodmap:

Modify keymaps and pointer button mappings in X11.

xev:

Display X events.