Character Data

A huge number of character representations (encodings) exist — but you need know only two:

- ASCII (ISO 646)
  - 7-bit values, using lower 7-bits of a byte (top bit always zero)
  - can encode roman alphabet, digits, punctuation, control chars
- UTF-8 (Unicode)
  - 8-bit values, with ability to extend to multi-byte values
  - can encode all human languages plus other symbols, e.g.:
    √ ∑ ∀ ∃
    🤑 🤗 🤓 😎 🤡 🤠

ASCII Character Encoding

- Uses values in the range `0x00` to `0x7F` (0..127)
- Characters partitioned into sequential groups
  - control characters (0..31) ... e.g. `'\0'`, `'\n'`
  - punctuation chars (32..47,91..96,123..126)
  - digits (48..57) ... `'0'..'9'`
  - upper case alphabetic (65..90) ... `'A'..'Z'`
  - lower case alphabetic (97..122) ... `'a'..'z'`
- Sequential nature of groups allow ordination e.g.
  `'3' - '0' == 3` `'J' - 'A' == 10`
- See `man 7 ascii`

Unicode

Widely-used standard for expressing "writing systems"

- not all writing systems use a small set of discrete symbols

Basically, a 32-bit representation of a wide range of symbols

- around 140K symbols, covering 140 different languages

Using 32-bits for *every* symbol would be too expensive

- e.g. standard roman alphabet + punctuation needs only 7-bits
- Several Unicode encodings have been developed
- UTF-8 most widely used encoding, dominates web-use
- UTF-8 clever encoding designed by Ken Thompson on restaurant napkin

UTF-8 Encoding

#bytes	*#bits*	*Byte 1*	*Byte 2*	*Byte 3*	*Byte 4*
1	7	`0xxxxxxx`	-	-	-
2	11	`110xxxxx`	`10xxxxxx`	-	-
3	16	`1110xxxx`	`10xxxxxx`	`10xxxxxx`	-
4	21	`11110xxx`	`10xxxxxx`	`10xxxxxx`	`10xxxxxx`

- The 127 1-byte codes are compatible with ASCII
- The 2048 2-byte codes include most Latin-script alphabets
- The 65536 3-byte codes include most Asian languages
- The 2097152 4-byte codes include symbols and emojis and ...

ch	*unicode*	*binary*	*UTF-8 encoding*
$	U+0024	**0100100**	0**0100100**
	U+00A2	**00010100010**	110**00010** 10**100010**
	U+20AC	**0010000010101100**	1110**0010** 10**000010** 10**101100**

UTF-8 Properties

- Compact, but not minimal encoding; encoding allows you to resync immediately if bytes lost from a stream.
- ASCII is a subset of UTF-8 - complete backwards compatibility!
- All other UTF-8 bytes $>$ 127 (0x7f).
- No byte of multi-byte UTF-8 encoding is 0 — can still use null-terminated strings.
- No byte of multi-byte UTF-8 encoding is valid ASCII.
- 0x2F (ASCII /) can not appear in multi-byte character — hence can use UTF-8 for Linux/Unix filename.
- C programs can treat UTF-8 similarly to ASCII.
- Beware: number of bytes in UTF-8 string != number of characters.