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.:
    \[\sqrt{\sum \forall \exists}\]
    😄 😊 😏 😎 😍 😎 😏
• 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
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 encodings have been developed (e.g. UTF-8)
UTF-8 Character Encoding

UTF-8 uses a variable-length encoding:

<table>
<thead>
<tr>
<th>#bytes</th>
<th>#bits</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>0xxxxxxx</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>110xxxxx</td>
<td>10xxxxxx</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>1110xxxxx</td>
<td>10xxxxxx</td>
<td>10xxxxxx</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>11110xxx</td>
<td>10xxxxxx</td>
<td>10xxxxxx</td>
<td>10xxxxxx</td>
</tr>
</tbody>
</table>

• 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 ...

Examples:

<table>
<thead>
<tr>
<th>ch</th>
<th>unicode</th>
<th>binary</th>
<th>UTF-8 encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>U+0024</td>
<td>0100100</td>
<td>00100100</td>
</tr>
<tr>
<td></td>
<td>U+00A2</td>
<td>00010100010</td>
<td>11000010 10100010</td>
</tr>
<tr>
<td></td>
<td>U+20AC</td>
<td>001000010101100</td>
<td>11100010 10000010 10101100</td>
</tr>
</tbody>
</table>
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 $\geq 127$ (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 $\neq$ number of characters.