Den endelige løsning? – Unicode og ISO 10646

- 21 biter, med mulighet for 1 114 112 kodepunkter
  - Ca 130 000 private
  - Ca 870 000 ennå ikke brukt
- Første 128 tegn er identisk med ASCII
- Første 256 tegn identisk med ISO 8859-1
- For hvert tegn finnes
  - en representativ glyf
  - kodepunktet
  - et navn
  - klassifisering
  - skriveretning

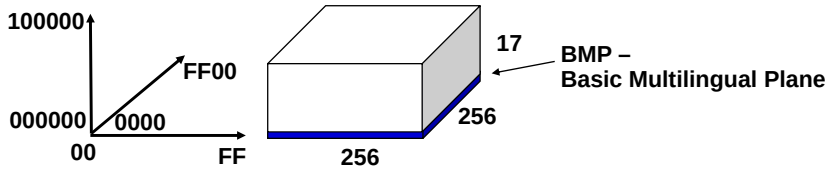
se http://www.unicode.org/charts/

- Vedtatte tegn med kodepunkter skal aldri endres

Unicode-kuben

- Tegnsettet er delt opp i 17 plan med 2^{16} = 65536 bitmønstre i hvert plan
- Plan 0: BMP – Basic Multilingual Plane U+0000 to U+FFFF
- Plan 1: SMP – Supplementary Multilingual Plane – historiske språk (f. eks. egyptiske hieroglyfer), musikk
- Plan 2: SIP – Supplementary Ideographic Plane – sjeldne kinesiske tegn
- Plan 14: SPP – Supplementary Special Purpose Plane – tag characters

I Unicode skriver vi U+ istedenfor 0x



Kodepunkt vs representasjon

- Kodepunkt:
  - Tegnets ”numeriske” verdi.
  - Det som står i kodetabellen.
- Representasjon:
  - Hvordan tegnet representeres som et bitmønster.

Unicode Transformation Formats

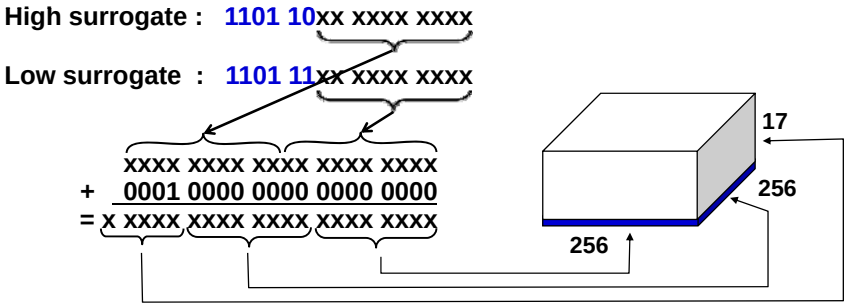
- UTF-32 (lite brukt):
  - Bruker 32 biter for alle tegn.
- UTF-16:
  - Tegn i BMP: 16 biter.
  - Tegn utenfor BMP: surrogatpar (32 biter).
- UTF-8:
  - Bruker 8, 16, 24, eller 32 biter avhengig av tegnet.
  - Tegnene i ASCII: 8 biter med ledende 0.

Unicode UTF-32

- Føy til ledende 0-biter opp til 32 biter foran representasjonen av kodepunktet
- Hva er representasjonen for A i UTF-32?
  (Kodepunktet for A er U+0041 = 0000 0000 0100 0001_2)
- Hva er representasjonen for ♘ i UTF-32?
  (Kodepunktet for ♘ er U+2658 = 0010 0110 0101 1000_2)

Unicode UTF-16

- Med 16 biter kan kodepunktene i BMP representeres direkte.
- De 16 planene over BMP adresseres med *surrogatpar*:
  - High surrogate – D800 – DBFF : 1101 10xx xxxx xxxx
  - Low surrogate – DC00 – DFFF: 1101 11xx xxxx xxxx
- Gir indirekte tilgang til 2^{20} = 1 048 576 ekstra tegn



Unicode UTF-16 – eksempel

- Hva er representasjonen for 𝄞 i Unicode UTF-16?
  (Kodepunktet for 𝄞 er U+1D11E = 1 1101 0001 0001 1110_2)

Unicode UTF-8

- Koding med variabel lengde
  - 8, 16, 24 eller 32 biter avhengig av kodepunktet

Unicode UTF-8 – eksempler

- Hva er representasjonen for Ø i Unicode UTF-8?
  (Kodepunktet for Ø er U+00D8 = 0000 0000 1101 1000_2)
- Hva er representasjonen for  i Unicode UTF-8?
  (Kodepunktet for er U+2658 = 0010 0110 0101 1000_2)
- Hva er representasjonen for  i Unicode UTF-16?
  (Kodepunktet for er U+1D11E = 1 1101 0001 0001 1110_2)

Big endian vs. Little endian

- I representasjoner som krever mer enn én byte, finnes det to mulige rekkefølger av bytene:
  - Starte med den mest signifikante (”Big endian”)
  - Starte med den minst signifikante (”Little/small endian”)
- Eksempel:
  UTF-16 Big endian for A er 0x 00 41
  UTF-16 Little endian for A er 0x 41 00
- Begge muligheter blir brukt i praksis, og dette kan gi problemer når data overføres fra et maskinmiljø til et annet!

Byte order mark (BOM)

- Et ”Byte order mark” (BOM) er tegnet ”Zero width no-break space” med kodepunkt U+FEFF i begynnelsen av en Unicode-fil.
- Siden det ikke finnes noe tegn med kodepunkt FFFE, kan BOM brukes til å finne filformatet (UTF-32, UTF-16, UTF-8 og Big eller Small endian):

Koding	BOM-bitmønster
UTF-32, big-endian	0x 00 00 FE FF
UTF-32, little-endian	0x FF FE 00 00
UTF-16, big-endian	0x FE FF
UTF-16, little-endian	0x FF FE
UTF-8	0x EF BB BF