

Character encoding

Victor Eijkhout

Notes for CS 594 – Fall 2004

Prehistory: Ascii

Ascii vs Ebcdic

- ▶ ASCII good, EBCDIC IBM[^]H[^]Hbad

Ascii vs Ebcdic

- ▶ ASCII good, EBCDIC IBM[^]H[^]Hbad
- ▶ ASCII: alphabet contiguous; EBCDIC not

Ascii vs Ebcdic

- ▶ ASCII good, EBCDIC IBM[^]H[^]Hbad
- ▶ ASCII: alphabet contiguous; EBCDIC not
- ▶ ASCII has unprintable or ‘control codes’

Ascii vs Ebcdic

- ▶ ASCII good, EBCDIC IBM[^]H[^]Hbad
- ▶ ASCII: alphabet contiguous; EBCDIC not
- ▶ ASCII has unprintable or ‘control codes’
- ▶ High bit always off

ISO 646, ASCII

20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
,	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
p	q	r	s	t	u	v	w	x	y	z	{		}	~	

'8-bit ASCII

- ▶ Languages other than English need accents
- ▶ Some languages need completely different alphabets

'8-bit ASCII

- ▶ Languages other than English need accents
- ▶ Some languages need completely different alphabets
- ▶ Code page: way of using the characters over 128
- ▶ Standards ISO 646-DE et cetera.

Dos: 437

	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	␣	␣	♥	♦	♣	♠	•	◻	◻	◻	◻	♀	♂	♂	♂
10	▶	◀	‡	!!	¶	§	■	‡	↑	↓	→	←	↵	↗	▼
20	!	!	!	#	\$	%	&	'	()	*	+	,	-	.
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N
50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	_
60	ˆ	a	b	c	d	e	f	g	h	i	j	k	l	m	n
70	p	q	r	s	t	u	v	w	x	y	z	{		}	~
80	Ç	ü	é	â	ä	à	ç	ê	ë	è	ì	î	ï	À	Á
90	Ê	æ	œ	ô	ö	ó	ú	û	ü	Ö	Ü	Φ	£	¥	ƒ
A0	ã	î	ô	û	ñ	Ñ	ª	º	¿	¬	¬	¥	¥	ì	«
B0	⌘	⌘	⌘		†	†	†	†	†	†	†	†	†	†	†
C0	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł
D0	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł	Ł
E0	α	β	γ	π	σ	μ	τ	ϑ	θ	Ω	δ	∞	∞	ε	η
F0	■	±	≥	≤	ƒ	ƒ	÷	≈	•	•	√	n	2	■	■

Microsoft Windows Latin 1

80	€		82	,	f	84	,,	85	...	86	†	87	‡	88	^	89	%	8A	Š	8B	<	8C	Œ		8E	Ž			
	91	‘	92	’	93	“	94	”	95	•	96	-	97	-	98	~	99	™	9A	š	9B	>	9C	œ		9E	ž	9F	ÿ
A0		i	A2	φ	£	℥	¥	!	§	A8	..	A9	©	AA	≡	AB	«	AC	¬	AD	-	AE	®	AF	-				
B0	°	±	B2	²	³	B4	⁴	B5	⁵	B6	⁶	B7	⁷	B8	˙	B9	₁	BA	₂	BB	»	BC	¼	BD	½	BE	¾	BF	¿
C0	À	Á	C2	Â	Ã	C4	Ä	C5	Å	C6	Æ	C7	Ç	C8	È	C9	É	CA	Ê	CB	Ë	CC	Ì	CD	Í	CE	Î	CF	Ï
D0	Ð	Ñ	D2	Ò	Ó	D4	Ô	D5	Õ	D6	Ö	D7	×	D8	Ø	D9	Ù	DA	Ú	DB	Û	DC	Ü	DD	Ý	DE	Þ	DF	ß
E0	à	á	E2	â	ã	E4	ä	E5	å	E6	æ	E7	ç	E8	è	E9	é	EA	ê	EB	ë	EC	ì	ED	í	EE	î	EF	ï
F0	ð	ñ	F2	ò	ó	F4	ô	F5	õ	F6	ö	F7	÷	F8	ø	F9	ù	FA	ú	FB	û	FC	ü	FD	ý	FE	þ	FF	ÿ

Recent history: ISO 8859

ISO 8859

- ▶ Set of standards: 1 for Latin, 2 east European, 5 Cyrillic
- ▶ first 32 positions over 128 left open for vendor extension

8859-1: Latin 1

A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	
	ı	ϕ	£	℥	¥	ı	§	¨	©	≡	«	¬	–	
B0	°	±	²	³	˘	μ	¶	·	¸	¹	º	»	¼	½
C0	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í
D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý
E0	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í
F0	ä	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý

Still trouble

- ▶ Asian alphabets can be really large
- ▶ DBCS: Double Byte Character Set

Still trouble

- ▶ Asian alphabets can be really large
- ▶ DBCS: Double Byte Character Set
- ▶ fun to code: no longer s++ and such

Unicode

ISO 10646

- ▶ UCS: Universal Character Set (ISO 10646)
- ▶ Unicode adds lots of goodies
- ▶ Over a million positions

Subsets

- ▶ First 128: ASCII
- ▶ First 256: ISO 8859-1
- ▶ First 2-byte plane: BMP (Basic Multilingual Plane)

Encodings

The idea of encoding

- ▶ Unicode is a list: how do you access elements?
- ▶ Six bytes is a waste for plain ASCII, incompatible
- ▶ Encodings of subsets, encodings for the whole caboodle

Examples

- ▶ UCS-2: two byte
- ▶ UTF-16: BMP
- ▶ UTF-8: one byte access to every character
- ▶ UTF-7: id, but based on 0–127 positions

UTF-8

- ▶ 0–127 rendered ‘as such’
- ▶ higher positions take 2–6 bytes:
 - ▶ first byte in the range 0xC0–0xFD (192–252)
 - ▶ next up to 5 bytes in the range 0x80–0xBF (128–191)
 - ▶ 8 = 1000 and B = 1011, so bit pattern starting with 10)
 - ▶ ⇒ six bits left for encoding
- ▶ (UTF-8 is standardized as RFC 3629.)

U-00000000 - U-0000007F	7 bits	0xxxxxxx	
U-00000080 - U-000007FF	$11 = 5 + 6$	110xxxxx	10xxxxxx
U-00000800 - U-0000FFFF	$16 = 4 + 2 \times 6$	1110xxxx	10xxxxxx 10xxxxxx
U-00010000 - U-001FFFFF	$21 = 3 + 3 \times 6$	11110xxx	10xxxxxx (3 times)
U-00200000 - U-03FFFFFF	$26 = 2 + 4 \times 6$	111110xx	10xxxxxx (4 times)
U-04000000 - U-7FFFFFFF	$31 = 1 + 5 \times 6$	1111110x	10xxxxxx (5 times)

Remaining stuff

Character sets

- ▶ Mapping from sequences of bytes to characters
- ▶ (reverse may not be unique)
- ▶ ISO 2022 defines switching between character sets; escape sequences

Character ↔ encoding

- ▶ Abstract Character Repertoire: list, unordered

Character ↔ encoding

- ▶ Abstract Character Repertoire: list, unordered
- ▶ Coded Character Set (code page, code points): numbers assigned

Character ↔ encoding

- ▶ Abstract Character Repertoire: list, unordered
- ▶ Coded Character Set (code page, code points): numbers assigned
- ▶ Character Encoding Form: mapping number to sequences of code units (UCS-2: one 16-bit unit, UTF-8: several 8-bit units)

Character ↔ encoding

- ▶ Abstract Character Repertoire: list, unordered
- ▶ Coded Character Set (code page, code points): numbers assigned
- ▶ Character Encoding Form: mapping number to sequences of code units (UCS-2: one 16-bit unit, UTF-8: several 8-bit units)
- ▶ Character Encoding Scheme: mapping to sequence of bytes (byte order, escape sequences)

Bootstrap problem

- ▶ How do you tell the receiver what character set you are using?
- ▶ MIBenum (RFC 1759, RFC 3808): unique names and numbers (IANA)
- ▶ Example: `ascii`

name `ANSI_X3.4-1968`

reference `RFC1345,KXS2`

MIBenum `3`

source `ECMA registry`

aliases `iso-ir-6, ANSI_X3.4-1986, ISO_646.irv:1991, ASCII, ISO646-US, US-ASCII (preferred MIME name), us, IBM367, cp367, csASCII`

Character sets in HTML

- ▶ Decimal or hex numerical code: ` `;
- ▶ Symbolic name: `©` is the copyright symbol.
- ▶ Use UTF-8 encoding; server states

Content-type: `text/html; charset=utf-8`

or file starts with

```
<META HTTP-EQUIV="Content-Type"  
      CONTENT="text/html; charset=utf-8">
```


More places

- ▶ Ftp: knows nothing, depends on client (line end, code page translation)
- ▶ Email: mime encoding
- ▶ Editors: emacs supports UTF-8
- ▶ Programming languages: Windows NT/2000/XP, (inc Visual Basic), uses UCS-2 natively:
Strings are declared `wchar_t` instead of `char`,
use `wcslen` instead of `strlen`
string is created as `L"Hello world"`.

Character issues in T_EX / L^AT_EX

Diacritics / accents

- ▶ original T_EX: accent placed over character; raised/lowered, shifted to center, extra shift for italic
⇒ because of shifts, no hyphenation possible
- ▶ 8-bit support in T_EX: possibility of fonts with accents, directly addressed

Old problems revisited

- ▶ Input file allows 8-bit: dependence on code page
`\usepackage[code]{applemac}`
- ▶ (unprintable characters made active: definition adjusted dynamically)
- ▶ Dependence on font organization left:
`\usepackage[T1]{fontenc}`