Character encoding

Victor Eijkhout

Notes for CS 594 - Fall 2004

Prehistory: Ascii Code pages Recent history: ISO 8859

Prehistory: Ascii

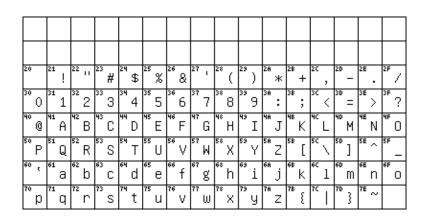
► ASCII good, EBCDIC IBM^H^H^Hbad

- ► ASCII good, EBCDIC IBM^H^H^Hbad
- ► ASCII: alphabet contiguous; EBCDIC not

- ► ASCII good, EBCDIC IBM^H^H^Hbad
- ► ASCII: alphabet contiguous; EBCDIC not
- AscII has unprintable or 'control codes'

- ► ASCII good, EBCDIC IBM^H^H^Hbad
- ► ASCII: alphabet contiguous; EBCDIC not
- Ascii has unprintable or 'control codes'
- High bit always off

ISO 646, ASCII



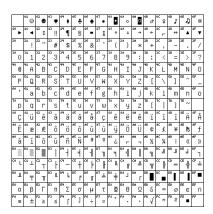
'8-bit Ascii

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

'8-bit Ascii

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

Dos: 437



Microsoft Windows Latin 1

80	€			82	,	83	f	84	,,	85		86	‡	87	#	88	^	89	%	8A	š	88	<	*C	Œ			\$E	ž	
		91	٢	92	,	93	cc	94	"	95	•	96	-	97	-	98	~	99	тн	9A	š	9B	>	9C	œ			9E	ž	9F Y
AO		A1	i	A2	¢	A3	£	A4	Ħ	A5	¥	A6	;	A7	S	A8		A9	0	AA	a	AB	«	AC	_	AD	-	AE	®	AF _
BO	۰	B1	±	B2	2	В3	3	вч	-	B5	μ	B6	1	B7		B8	,	В9	1	BA	0	BB	>>	BC	14	BD	K	BE	¥	BF خ
CO	À	C1	Á	CZ	Â	C3	Ã	СЧ	Ä	C5	Å	C6	Æ	C7	Ç	C8	È	C9	É	CA	Ê	СВ	Ë	СС	Ì	CD	Í	CE	Î	CF ::
DO	Đ	D1	Ñ	DZ	ò	D3	Ó	D4	ô	D5	õ	D6	ö	D7	×	D8	Ø	D9	Ù	DA	Ú	DB	Û	DC	Ü	DD	Ý	DE	Þ	₽₽В
ΕO	à	E1	á	E2	â	E 3	ã	EЧ	ä	E5	å	E6	æ	E7	Ç	E8	è	E9	é	EΑ	ê	EB	ë	EC	ì	ED	í	EE	î	EF 1
FO	ð	F1	ñ	F2	ò	F3	ó	F٤	ô	F5	õ	F6	ö	F7	÷	F8	Ø	F9	ũ	FA	ú	FB	û	FC	ü	FD	ý	FE	Þ	FF ;;

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

AO		A1	i	AZ	¢	A3	£	A4	Ħ	A5	¥	A6	;	A7	8	A8		A9	0	AA	a	AB	((AC	7	AD	_
BO	۰	B1	±	BZ	2	B3	3	вч	-	B5	μ	B6	1	В7		B8	,	B9	1	BA	0	BB	>>	BC	14	BD	K
CO	À	C1	Á	C2	Â	C3	Ã	СЧ	Ä	C5	Å	Ce	Æ	C7	Ç	C8	È	C9	É	CA	Ê	СВ	Ë	СС	Ì	CD	Í
DO	Đ	D1	Ñ	DZ	ò	D3	Ó	D4	ô	D5	õ	De	ö	D7	×	D8	Ø	D9	Ù	DA	Ú	DB	Û	DC	Ü	DD	Ý
ΕO	à	E1	á	E2	â	E 3	ã	EЧ	ä	E5	å	E6	æ	E7	Ç	E\$	è	E9	é	ΕA	ê	EB	ë	EC	ì	ED	í
FO	ð	F1	ñ	F2	ò	F3	ó	F4	ô	F5	õ	F6	ö	F7	÷	F8	Ø	F9	ù	FA	ú	FB	û	FC	ü	FD	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)
 - \triangleright 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-7FFFFFF	$31 = 1 + 5 \times 6$	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

$Character \leftrightarrow encoding$

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

$Character \leftrightarrow encoding$

- ▶ 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)

$Character \leftrightarrow encoding$

- ► 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)
- name ANSI_X3.4-1968

Example: ascii

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

Charactersets in HTML

- ▶ Decimal or hex numerical code:
- Symbolic name: © is the copyright symbol.

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 TEX / LATEX

Diacritics / accents

- original TEX: 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}