Introduction to the Use of Computers

Christophe Rhodes
c.rhodes@gold.ac.uk

Autumn 2012, Fridays: 10:00–12:00: WTA & 15:00–17:00: WHB 300

Files

Binary Files

Files can contain information which is not designed to be interpreted directly as text.
Examples:

- PDF documents (`25 50 44 46 2d 31 2e 34 ...`)

Files

Binary Files

Files can contain information which is not designed to be interpreted directly as text.
Examples:

- PDF documents (`25 50 44 46 2d 31 2e 34 ...`)
  `%PDF....`
- MIDI files (`4d 54 68 64 00 00 00 06 ...`)

Files

Binary Files

Files can contain information which is not designed to be interpreted directly as text.
Examples:

- PDF documents (`25 50 44 46 2d 31 2e 34 ...`)
  `%PDF....`
- MIDI files (`4d 54 68 64 00 00 00 06 ...`)
  `MThd....`

Files

Binary Files

Files can contain information which is not designed to be interpreted directly as text.
Examples:

- PDF documents (`25 50 44 46 2d 31 2e 34 ...`)
  `%PDF....`
- MIDI files (`4d 54 68 64 00 00 00 06 ...`)
  `MThd....`
- images (`89 50 4e 47 0d 0a 1a 0a ...`)

Files

Binary Files

Files can contain information which is not designed to be interpreted directly as text.
Examples:

- PDF documents (`25 50 44 46 2d 31 2e 34 ...`)
  `%PDF....`
- MIDI files (`4d 54 68 64 00 00 00 06 ...`)
  `MThd....`
- images (`89 50 4e 47 0d 0a 1a 0a ...`)
  `.PNG....`

Files

Binary Files

How to make sure that a binary file is not treated as text?

- PNG header: `89 50 4e 47 0d 0a 1a 0a ...`
- 89: top bit set (detect non-8-bit-clean systems)
- `50 4e 47`: `PNG` (identify file type)
- `0d 0a`: DOS-style newline (`CR LF`)
- `1a`: DOS-style end-of-file
- `0a`: Unix-style newline (`LF`)

Character Sets and Encoding

ASCII

ASCII printing characters:

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	
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Character Sets and Encodings

ASCII

Good things about ASCII:

- (almost) the only standard
  - EBCDIC
  - national variants of ASCII
- basically able to reliably communicate in English

Character Sets and Encodings

ASCII

Good things about ASCII:

- (almost) the only standard
  - EBCDIC
  - national variants of ASCII
- basically able to reliably communicate in English

Problems with ASCII:

- only able to reliably communicate in English...

Character Sets and Encodings

ASCII

Good things about ASCII:

- (almost) the only standard
  - EBCDIC
  - national variants of ASCII
- basically able to reliably communicate in English

Problems with ASCII:

- only able to reliably communicate in English...
- and even then, highly limited
  - *raison d'être*;
  - *naïveté*, *Noël*
  - *mañana*

Character Sets and Encodings

ASCII

Good things about ASCII:

- (almost) the only standard
  - EBCDIC
  - national variants of ASCII
- basically able to reliably communicate in English

Problems with ASCII:

- only able to reliably communicate in English...
- and even then, highly limited
  - *raison d'être*;
  - *naïveté*, *Noël*
  - *mañana*
  - *You owe me £5*

Character Sets and Encodings

ASCII

Obvious solution:

- Let's invent a character set which can represent every character in the world...

Character Sets and Encodings

ASCII

Obvious solution:

- Let's invent a character set which can represent every character in the world...
- and let's invent encoding methods so that it's trivial to encode and decode them.

Character Sets and Encodings

ASCII

Obvious solution:

- Let's invent a character set which can represent every character in the world...
- and let's invent encoding methods so that it's trivial to encode and decode them.

If only things had happened like that...

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Instead:

- ASCII is a 7-bit code;
- most bytes are 8-bits;
- ...

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Instead:

- ASCII is a 7-bit code;
- most bytes are 8-bits;
- ...
- Profit!

ISO-8859-x

- use the ‘spare’ 128 code points
- cram in characters for as many languages as possible
- multiple variants for mulitple geographic and linguistic regions

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ
- ISO-8859-3: Southern Europe

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ
- ISO-8859-3: Southern Europe: ĝ, İ

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ
- ISO-8859-3: Southern Europe: ĝ, İ
- ISO-8859-4: Northern Europe

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ
- ISO-8859-3: Southern Europe: ĝ, İ
- ISO-8859-4: Northern Europe: Ո, ☒

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ
- ISO-8859-3: Southern Europe: ĝ, İ
- ISO-8859-4: Northern Europe: Ո, ☒
- ISO-8859-5: Cyrillic

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ
- ISO-8859-3: Southern Europe: ĝ, İ
- ISO-8859-4: Northern Europe: Ŋ, ⊠
- ISO-8859-5: Cyrillic: Д, Ж, щ

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ
- ISO-8859-3: Southern Europe: ĝ, İ
- ISO-8859-4: Northern Europe: Ŋ, ☒
- ISO-8859-5: Cyrillic: Д, Ж, щ
- ISO-8859-6: Arabic

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ
- ISO-8859-3: Southern Europe: ĝ, İ
- ISO-8859-4: Northern Europe: Ŋ, ☒
- ISO-8859-5: Cyrillic: Д, Ж, щ
- ISO-8859-6: Arabic: ,☒ ☒

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ
- ISO-8859-3: Southern Europe: ĝ, İ
- ISO-8859-4: Northern Europe: Ŋ, ☒
- ISO-8859-5: Cyrillic: Д, Ж, щ
- ISO-8859-6: Arabic: ,☒ ☒
- ... when will the madness stop? ...

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ
- ISO-8859-3: Southern Europe: ĝ, İ
- ISO-8859-4: Northern Europe: Ŋ, ☒
- ISO-8859-5: Cyrillic: Д, Ж, щ
- ISO-8859-6: Arabic: ,☒ ☒
- ... when will the madness stop? ...
- ISO-8859-15: Western Europe (now with Euro Sign!)

Character Sets and Encodings

The ISO-8859 Alphabet Soup

Mid-1980s (finalized 1998):

- ISO-8859-1: Western Europe: é, è, £, ñ, ß, ÿ
- ISO-8859-2: Central Europe: š, ş, ŕ, õ
- ISO-8859-3: Southern Europe: ĝ, İ
- ISO-8859-4: Northern Europe: Ŋ, ☒
- ISO-8859-5: Cyrillic: Д, Ж, щ
- ISO-8859-6: Arabic: ,☒ ☒
- ... when will the madness stop? ...
- ISO-8859-15: Western Europe (now with Euro Sign!)
- ISO-8859-16: South-Eastern Europe (now with Euro Sign!)

Cannot even begin to address issue of ideographic languages

Character Sets and Encodings

Unicode

Early 1990s (not finalized yet):

- the ‘right’ solution:
  - aim to collect all characters in use
  - what is a character, anyway?

Character Sets and Encodings

Unicode

Early 1990s (not finalized yet):

- the 'right' solution:
  - aim to collect all characters in use
  - what is a character, anyway?
- all characters given a unique, unchangeable codepoint:
  - ASCII-compatible: all ASCII characters given their ASCII codepoint;
  - sort-of-ISO-8859-1-compatible: all ISO-8859-1 characters given their ISO-8859-1 codepoint (but encoding makes this complicated in practice);
  - not backwards-compatible with anything else.
  - examples:
    - 'a' is `U+0061`
    - '£' is `U+00A3`
    - '€' is `U+20AC`
  - more than 65536 characters in Unicode (space for almost 2097152)
- the job does not end there...

Character Sets and Encodings

Unicode

Given a sequence of characters, how to encode for storage or transmission?

- convert into a sequence of codepoints

Character Sets and Encodings

Unicode

Given a sequence of characters, how to encode for storage or transmission?

- convert into a sequence of codepoints
- ... then what?

UCS-2:

- 65536 characters will be enough for everybody...
- ... right?
- so just encode each code point in two 8-bit bytes (fixed-width encoding).

UTF-16 (Java, Windows):

- more than 65536 characters needed?
- encode rare ones as pairs of surrogate characters
- variable-width encoding

Character Sets and Encodings

Unicode

UCS-4 / UTF-32 (simple in-memory representation):

- 4294967296 characters will be enough for everybody...
- ... right?
- so just encode each code point in four 8-bit bytes (fixed-width encoding)

Problems with UCS-4:

- encoding not backwards-compatible with ASCII;
- very wasteful of space (factor of 4).

Character Sets and Encodings

Unicode

UTF-8 (Mac OS X, Unix):

- variable-length encoding;
- ASCII characters are encoded as their character code in a single byte;
- other characters are encoded as multiple bytes:
  - first byte encodes length as well as some code bytes
  - remaining 8-bit bytes of the form 10xxxxxx
- examples:
  - 'a' encoded as `61`
  - '£' encoded as `c2 a3`
  - '€' encoded as `e2 82 ac`