File Formats

DSCI 551

Wensheng Wu

File Formats

- Specify what information bits in file encode
- Example: text file
  - String of characters with particular encoding scheme, e.g., ASCII and Unicode
  - E.g., TXT, HTML, JSON, XML
- Others: xls, ppt, pdf, jpg, gif, mp3, png, etc.

Roadmap

- Character encoding 
  - ASCII
  - Unicode
- JSON (done earlier)
- XML (will talk about it next)

Code space & points

0001
0
4
6
6

- Code space
  - A range of numerical values available for encoding characters
  - E.g., 0 to 10FFFF for Unicode, 0 to 7F for ASCII

code space > code point > code unit

- Code point
  - A value for a character in a code space

- Unicode code point
  - U+ followed by its hexadecimal value, e.g., U+0058 for capital letter ‘X’) => u'\u0058'

Encoding (of code points)

- Code unit: the smallest unit (comprising a number of bits) used to construct an encoding for a code point
  - Code unit for UTF-8: 8-bit
  - UTF-16:16-bit

- UTF (Unicode Transformation Format) encoding
  - E.g., UTF-8 and UTF-16

Variable-length encoding

- Characters encoded using codes of different length

- In Unicode, a code point may be represented using multiple code units
  - E.g., 1-4 in UTF-8, 1-2 in UTF-16

意思是1个到4个code units in UTF-8，就是1 * 8 bit 到 4 * 8 bit

1个到2个 code units in UTF-16，就是1 * 16 bit到 2 * 16 bit

ASCII

- American Standard Code for Information Interchange

- 128 characters: 7-bit code (code points: 0~7F)
  - Digits: 0-9 (0x30 – 0x39)
  - Uppercase letters: A-Z (0x41 – 0x5A)
  - Lowercase letters: a-z (0x61 – 0x7A)
  - White space (0x20)
  - Punctuation symbols
  - Control characters (e.g., Ctrl-C: 0x03)

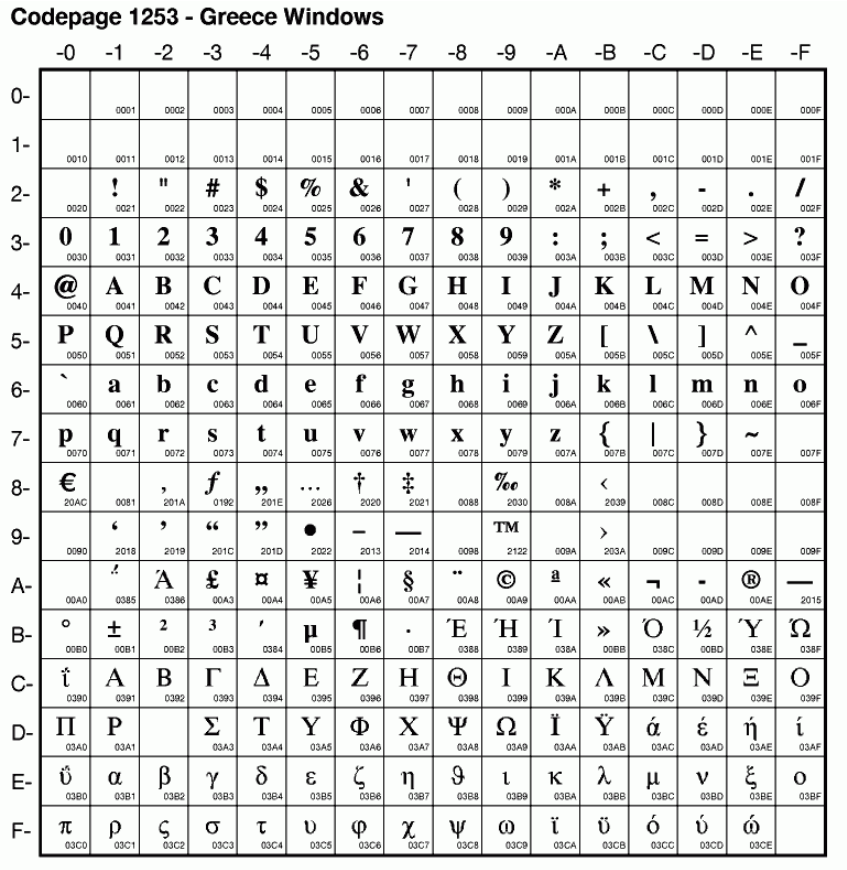
1111 = 15 = F

ASCII

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Windows-1253

- Windows code page for Latin + Greek characters
- Use 8 bits
  - 0x00 ~ 0xFF



Unicode

- Unicode supports more characters than ASCII and various codepages
  what is a codepage?
  e.g. ASCII, Windows code page for Latin+Greek
- Unicode separates code points from encoding
  - In contrast to ASCII, where code point = encoding

最好都记住

Unicode

- Code space is divided into 17 planes
- Each plane = contiguous 2^{16} code points
- Recall that code points range from 0 to 10FFFF

$\Rightarrow$Total code points = 17 * 2^{16} or 1,114,112 code points

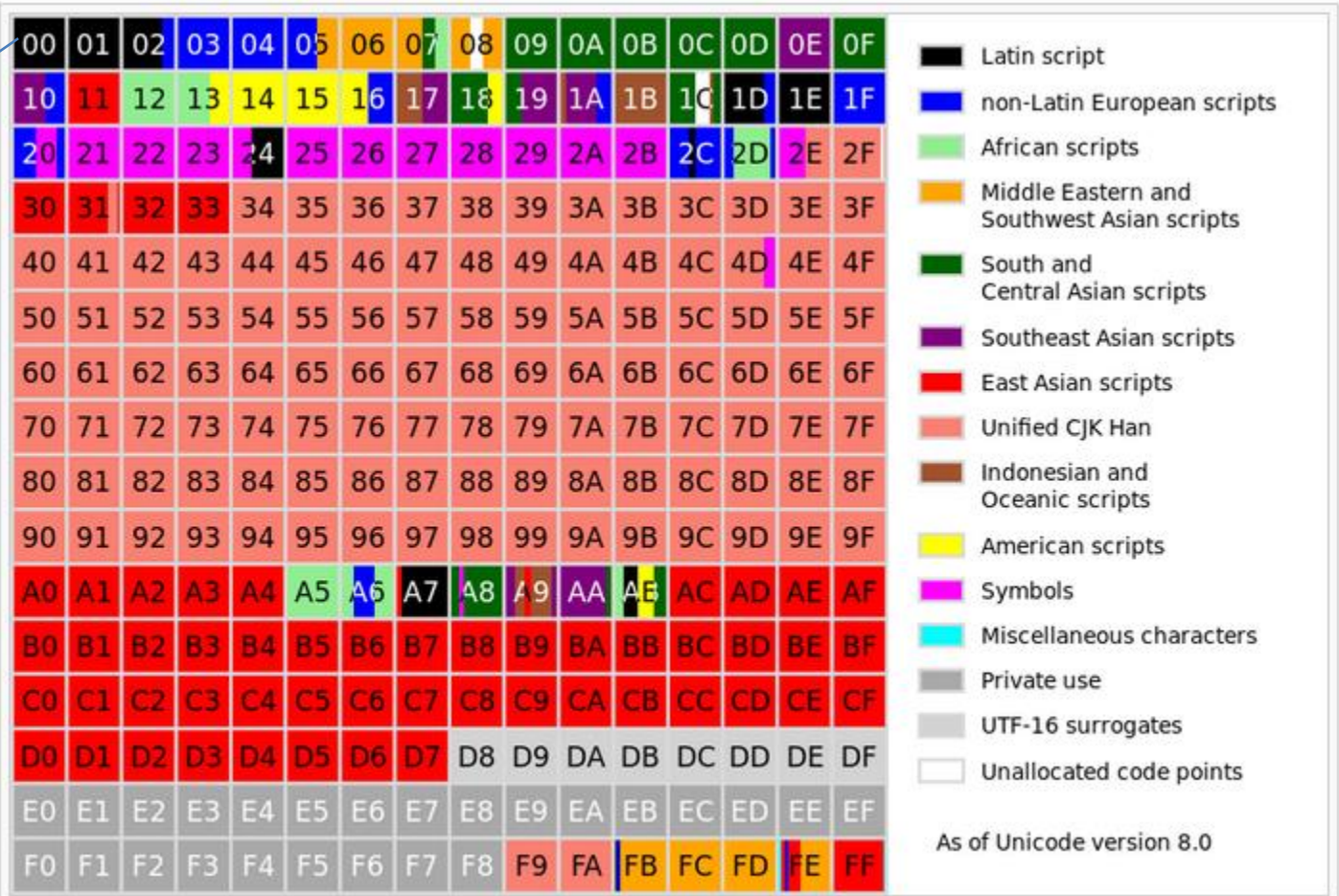
Note 2^{16} = 65,536

Planes in Unicode

V·T·E	Unicode planes and used code point ranges											[hide]
Basic		Supplementary										
Plane 0		Plane 1		Plane 2		Planes 3–13		Plane 14		Planes 15–16		
0000–FFFF		10000–1FFFF		20000–2FFFF		30000–DFFFF		E0000–EFFFF		F0000–10FFFF		
Basic Multilingual Plane		Supplementary Multilingual Plane		Supplementary Ideographic Plane		*unassigned*		Supplement-ary Special-purpose Plane		Supplement-ary Private Use Area		
BMP		SMP		SIP		—		SSP		S PUA A/B		
0000–0FFF	8000–8FFF	10000–10FFF		20000–20FFF	28000–28FFF			E0000–E0FFF		*15: PUA-A* F0000–FFFFF		
1000–1FFF	9000–9FFF	11000–11FFF		21000–21FFF	29000–29FFF							
2000–2FFF	A000–AFFF	12000–12FFF		22000–22FFF	2A000–2AFFF							
3000–3FFF	B000–BFFF	13000–13FFF	1B000–1BFFF	23000–23FFF	2B000–2BFFF					*16: PUA-B* 100000–10FFFF		
4000–4FFF	C000–CFFF	14000–14FFF		24000–24FFF	2C000–2CFFF							
5000–5FFF	D000–DFFF		1D000–1DFFF	25000–25FFF								
6000–6FFF	E000–EFFF	16000–16FFF	1E000–1EFFF	26000–26FFF								
7000–7FFF	F000–FFFF		1F000–1FFFF	27000–27FFF	2F000–2FFFF							

Plane 0: BMP (Basic Multilingual Plane)

Block 00
Represents
0000~00FF



A map of the Basic Multilingual Plane. Each numbered box represents 256 code points.

Each block represents 256 code points

UTF-8

- Encoding scheme for Unicode code space
- Code unit = 8 bits
- Variable length
  - Code point may be represented using 1-4 code units

UTF-8 Design

- ASCII characters use one code unit
  - First bit is zero
- Other Unicode characters use up to 4 units

lecture 01/31 2:20:00左右的视频

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

UTF-8 Features

lecture 01/31 2:20:00左右的视频

- Backward compatibility
  - One byte for ASCII, leading bit of byte is zero
- Clear distinction btw single- vs. multi-byte characters
  - Single-byte/multi-byte: start with 0/1 respectively
- Multiple length
  - a leading byte starts with 2 or more 1's, followed by a 0, e.g., '110', '1110', etc.
  - One or more continuation bytes all start with '10'

UTF-8 Features

- Clear indication of code sequence length
  - By # of 1's in leading byte (for multi-byte)

- Self-synchronization
  - Can find start of characters by backing up at most 3 bytes

最多向前（左）追溯3byte就能找到开头

Example

- Encode ‘€’ using UTF-8
- Code point = U+20AC
  - 10 0000 1010 1100
- Need 3 bytes in UTF-8

Character		Binary code point	Binary UTF-8	Hexadecimal UTF-8
$	U+0024	0100100	00100100	24
¢	U+00A2	00010100010	11000010 10100010	C2 A2
€	U+20AC	0010000010101100	11100010 10000010 10101100	E2 82 AC
𐍈	U+10348	00001000000110100 1000	11110000 10010000 10001101 10001000	F0 90 8D 88

Unicode in Python

```
>>> a = '\u20ac'
>>> a
'€'

>>> e = a.encode('utf-8')
>>> e
b'\xe2\x82\xac'

>>> e.decode('utf-8')
'€'
```

Resources

- UTF-8
  - https://en.wikipedia.org/wiki/UTF-8
- UTF-16
  - https://en.wikipedia.org/wiki/UTF-16