Лекция 10. Кодировки текста

Александр Смаль

CS центр
15 апреля 2013
Санкт-Петербург

Введение

- Набор символов (character set) — таблица, задающая кодировку конечного множества символов алфавита (обычно элементов текста: букв, цифр, знаков препинания).
- Такая таблица сопоставляет каждому символу последовательность длиной в один или несколько символов другого алфавита (точек и тире в коде Морзе, сигнальных флагов на флоте, нулей и единиц (битов) в компьютере).
- В настоящее время в основном используются кодировки трёх типов: совместимые с ASCII, совместимые с EBCDIC и основанные на Юникоде 16-битные, с подавляющим преобладанием первых.

BCDIC

- BCDIC (Binary Coded Decimal Interchange Code) — шестибитная кодировка, содержащая латинские символы, арабские цифры, знаки пунктуации и некоторые математические символы.
- Создана в IBM в 1928 году.
- Кодировка изначально адаптировалась для перфокарт.
- Было очень много вариантов этой кодировки, единого стандарта не существовало.
- EBCDIC (Extended BCDIC) — стандартный восьмибитный код, разработанный IBM для использования на мэйнфреймах IBM и совместимых с ними.
- Существовало по меньшей мере шесть версий EBCDIC, несовместимых между собой.

ASCII

- ASCII (American Standard Code for Information Interchange) — американская стандартная кодировочная таблица для печатных символов и некоторых специальных кодов.
- Изначально разработанная как 7-битная.
- Сейчас воспринимается как половина 8-битной.
- Поддерживала наложение символов при помощи символа BS (аналогичный подход используется в man).
- Стандарт ISO 646 (ECMA-6) предусматривает возможность размещения национальных символов на месте `@ [ \ ] ^ ‘ { | } ~`.
- Это удобно для европейских языков, где нужны лишь несколько дополнительных символов. Вариант ASCII без национальных символов называется US-ASCII, или «International Reference Version».

ASCII

Нынешний ASCII стандартизован в 1986 году (ANSI X3.4, RFC 20, ISO/IEC 646:1991, ECMA-6) Американским национальным институтом по стандартизации (American National Standards Institute, ANSI).

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Расширения ASCII

- Для европейских языков, содержащих символы с акцентами и не латинскими буквами были разработаны стандарты семейства ISO 8859.
- Они были обратно совместимы с ASCII, но также использовали восьмой бит для дополнения таблицы дополнительными 127 символами для каждой кодировки.
- На данный момент существует 15 вариантов стандарта ISO 8859 (от 8859-1 до 8859-15).
- Для IBM PC с 1981 года использовалась кодировка CP437 (Codepage 437, DOSLatinUS). Кодировка аппаратно поддерживалась видеоконтроллерами MGA и VGA.
- CP437 послужила образцом для множества других кодировок, в том числе альтернативной кодировки для русского языка CP866.

Расширения ASCII (продолжение)

- При разработке Microsoft Windows был разработан набор символов Windows-1252. Этот набор является надстройкой ISO 8859-1, обладающий собственными изменениями.
- На основе Windows-1252 было разработано множество национальных кодировок. В т.ч. Windows-1251.
- В среде UNIX получили распространение семейство кодировок KOI для кириллических алфивитов: KOI8-R (русский, болгарский) и KOI8-U (украинский).
- В KOI8 фонетически созвучные кириллические символы расположены так же, как и латинские, благодаря чему даже при отбрасывании восьмого бита текст оставался читабельным в ASCII-терминалах в виде транслита.
- Для поддержки кириллических символов Apple разрабатывает собственную кодировку (MacCyrillic).

Не-ASCII кодировки

- В азиатских странах была необходимость разработки совершенно отличных от ASCII однобайтовых кодировок для нелатинских алфавитов.
- Например EUC (Extended Unix Coding), используемый в японском и корейском (и в меньшей степени китайском) алфавитах, породила еще большую неразбериху с кодировками, из-за чего ряд операционных систем все еще использует различные наборы символов для одного и того же языка, например японские Shift-JIS и ISO-2022-JP..

Предпосылки к появлению Unicode

- Множество конкурирующих стандартов.
- Все приложения (веб-браузеры, почтовые клиенты, текстовые редакторы) вынуждены уметь работать в множестве кодировок.
- Дублирование шрифтов: для каждой кодировки создавалась копия шрифта.
- Не понятно как писать текст на нескольких языках одновременно.
- Проблема совмещения текста с разным направлением.
- Множество алфавитов, которые ещё не имеют стандартов, а значит они бы скоро появились ◡̈.

Unicode

- Unicode — стандарт кодирования символов, позволяющий представить знаки практически всех письменных языков.
- Стандарт предложен в 1991 году некоммерческой организацией Unicode Consortium.
- Стандарт состоит из двух основных разделов: универсальный набор символов (UCS, universal character set) и семейство кодировок (UTF, Unicode transformation format).
- Универсальный набор символов задаёт однозначное соответствие символов кодам — элементам кодового пространства, представляющим неотрицательные целые числа. Семейство кодировок определяет машинное представление последовательности кодов UCS.

Устройство Unicode

- Коды в стандарте Unicode разделены на несколько областей.
- Область с кодами от U+0000 до U+007F содержит символы набора ASCII с соответствующими кодами.
- Далее расположены области знаков различных письменностей, знаки пунктуации и технические символы.
- Часть кодов зарезервирована для использования в будущем.
- Под символы кириллицы выделены области знаков с кодами от U+0400 до U+052F, от U+2DE0 до U+2DFF, от U+A640 до U+A69F.

Развитие Unicode

- Первая версия Юникода представляла собой кодировку с фиксированным размером символа в 16 бит, то есть общее число кодов было 2^{16}. (отсюда практика обозначения символов четырьмя шестнадцатеричными цифрами).
- При этом в Юникоде планировалось кодировать не все существующие символы, а только те, которые необходимы в повседневном обиходе. Редко используемые символы должны были размещаться в «области пользовательских символов» (private use area), которая первоначально занимала коды U+D800 – U+F8FF.
- Чтобы использовать Unicode в качестве промежуточного звена при преобразовании разных кодировок друг в друга, в него включили все символы, представленные во всех наиболее известных кодировках.

История

- В дальнейшем было принято решение кодировать все символы и в связи с этим расширить кодовую область.
- Поскольку в ряде компьютерных систем (например, Windows NT) фиксированные 16-битные символы уже использовались в качестве кодировки по умолчанию, было решено все наиболее важные знаки кодировать только в пределах первых 2^{16} позиций (так называемая basic multilingual plane, BMP).
- Остальное пространство используется для «дополнительных символов» (supplementary characters): систем письма вымерших языков или очень редко используемых китайских иероглифов, математических и музыкальных символов.

UTF-16

- Для совместимости со старыми 16-битными системами была изобретена система UTF-16, где первые 2^{16} позиций, за исключением позиций из интервала U+D800 – U+DFFF, отображаются непосредственно как 16-битные числа, а остальные представляются в виде «суррогатных пар» (первый элемент пары из области U+D800 – U+DBFF, второй элемент пары из области U+DC00 – U+DFFF).
- Для суррогатных пар была использована часть кодового пространства (2048 позиций), ранее отведённого для «символов для частного использования».
- Поскольку в UTF-16 можно отобразить только $2^{20} + 2^{16} - 2048 = 1{,}112{,}064$ символов, то это число и было выбрано в качестве окончательной величины кодового пространства Unicode.

История

- Хотя кодовая область Unicode была расширена за пределы 2^{16} уже в версии 2.0 (1993), первые символы в «верхней» области были размещены только в версии 3.1 (2001). Роль этой кодировки в веб-секторе постоянно растёт, на начало 2010 доля веб-сайтов, использующих Юникод, составила около 50%.

Плоскости Unicode

- В версии 6.0 используется чуть менее 110,000 кодовых позиций (109,242 графических и 273 прочих символов).
- Кодовое пространство разбито на 17 плоскостей по 2^{16} символов.
- Нулевая плоскость называется базовой, в ней расположены символы наиболее употребительных письменностей.
- Первая плоскость используется, в основном, для исторических письменностей.
- Вторая — для редко используемых китайских иероглифов.
- Третья зарезервирована для архаичных китайских иероглифов.
- Плоскости 15 и 16 выделены для частного употребления.

Недостатки Unicode

- В Юникоде английское «a» и польское «a» — один и тот же символ. Аналогично (но отличающимся от «a» латинского) считаются русское «a» и сербское «a».
- Есть возможность переключать вывод текста слева направо и в обратную сторону. Аналогичной функциональности для письма сверху вниз нет.
- Unicode предусматривает возможность разных начертаний одного и того же символа в зависимости от языка. Поэтому нужно следить, чтобы текст всегда был правильно помечен как относящийся к тому или другому языку.
- Перевод из строчных букв в заглавные зависит от языка.
- Файлы неанглийского текста в Юникоде всегда занимают больше места.
- Обработка Unicode текста медленнее.

Нормализация

- Один и тот же символ в Unicode может быть представим как монолитный и как составной (модификатор + симовол).
- К примеру, русские буквы Ё (U+0401) и Й (U+0419) существуют в виде монолитных символов, хотя могут быть представлены и набором базового символа с последующим диакритическим знаком, то есть в составной форме (decomposed): Е+¨ (U+0415 U+0308), И+˘ (U+0418 U+0306).
- Соответственно, один и тот же текст может быть записан по-разному. Это влияет, к примеру, на сравнение строк.
- Необходимо производить нормализацию Unicode текста.

Unicode transformation format

- UCS-2 — двухбайтовая кодировка, которая содержит первую плоскость Unicode.
- UTF-16 (UTF-16BE, UTF-16LE) — многобайтовая кодировка (2 или 4 байта на символ), позволяет отображать всю таблицу Unicode.
- UTF-32 (UTF-32BE, UTF-32LE), UCS-4 — четырёхбайтовая кодировка, позволяет отображать всю таблицу Unicode.
- UTF-8 — многобайтовая совместимая с ASCII кодировка (до 6 байт на символ), позволяет отображать всю Unicode.
- Была разработана также форма представления UTF-7, но из-за несовместимости с ASCII она не получила распространения и не включена в стандарт.
- 1 апреля 2005 года были предложены две шуточные формы представления: UTF-9 и UTF-18 (RFC 4042).

Порядок байтов

- В потоке данных UTF-16 старший байт может записываться либо перед младшим (little-endian), либо после младшего (big-endian). Аналогично проблема у четырёхбайтной кодировки UTF-32.
- Для определения формата представления в начало текстового файла записывается сигнатура, также именуемый меткой порядка байтов (byte order mark, BOM).

UTF-8	`EF BB BF`
UTF-16BE	`FE FF`
UTF-16LE	`FF FE`
UTF-32BE	`00 00 FE FF`
UTF-32LE	`FF FE 00 00`

- Файлы в кодировках UTF-16 и UTF-32, не содержащие BOM, должны иметь порядок байтов big-endian.

UTF-8

- UTF-8 — представление Юникода, обеспечивающее наилучшую совместимость со старыми системами, использовавшими 8-битные символы.
- В тексте UTF-8 любой байт со значением меньше 128 изображает символ ASCII с тем же кодом.
- Остальные символы Юникода изображаются последовательностями длиной от 2 до 6 байт (на деле, до 4 байт, поскольку нет символов с кодом больше `10FFFF`).
- В многобайтовых символах первый байт всегда имеет вид `11xxxxxx`, а остальные — `10xxxxxx`.
- Сейчас стандарт UTF-8 официально закреплён в документах RFC 3629 и ISO/IEC 10646 Annex D.

Представление Unicode в UTF-8

```
0x00000000-0x0000007F: 0xxxxxxx
0x00000080-0x000007FF: 110xxxxx 10xxxxxx
0x00000800-0x0000FFFF: 1110xxxx 10xxxxxx 10xxxxxx
0x00010000-0x001FFFFF: 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
```

Теоретически возможны, но не включены в стандарт также:

```
0x00200000-0x03FFFFFF:
    111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
0x04000000-0x7FFFFFFF:
    1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
```

UTF-8 позволяет указать один и тот же символ несколькими способами, но только наиболее короткий из них правильный. Остальные формы должны отвергаться по соображениям безопасности.

ISO/IEC 10646

- Консорциум Юникода работает в тесной связи с рабочей группой ISO/IEC/JTC1/SC2/WG2, которая занимается разработкой международного стандарта 10646 (ISO/IEC 10646).
- Между стандартом Юникода и ISO/IEC 10646 установлена синхронизация, хотя каждый стандарт использует свою терминологию и систему документации.

Как передавать данные по текстовым каналам

- В некоторых приложениях (к примеру, в электронной почте, URL-ах) ограничен набор возможных символов.
- В электронной почте используется кодирование BASE64. При кодировании байты разбиваются на кусочки по 6 бит и каждому из них сопоставляется один из 64 печатных символов (a-zA-Z0-9+/).
- Для того, чтобы передать произвольные символы в URL используется URL encoding (percent-encoding), когда символу сопоставляется его номер предварённый процентом, к примеру “%20” для символа “пробел”. Символы Unicode кодируются в UTF-8 несколькими байтами: “и” соответствует “%D0%B8”.

Программы для конвертации кодировок

- `iconv` — позволяет конвертировать текст из одной кодировки в другую.
- `uniconv` — позволяет конвертировать текст из одной кодировки в другую через преобразование в Unicode.
- `enca` — пытается “угадать” кодировку текста.
- `luit` — позволяет запускать терминальные приложения в другой кодировке.

Спасибо за внимание!