Текстовые кодировки

Про кодировку чисел - двубайтовую

Big Ending & little Ending – разные способы кодирования чисел (когда в поток кладется сначала старший или сначала младший байт)

В Ip TCP UDP используется big ending

Про кодировку текста - однобайтовую

ASCII один из стандартов кодировки – таблица кодов [0, 127]

Есть кодировка ISO стандарты кодировки для разных языков

Проблемы кодировок :

1. если текст хотим раскодировать, нужно знать в какой кодировке он находится
2. не можем все символы мира в одну записать кодировку. Если текст на разных языках, то нужно больше чем одну кодировку использовать.
3. Иногда даже в рамках одного языка не хватает одной кодировки (например для китайского языка, символов > 256)

Из-за этих проблем появилась международная организация, которая разрабатывает универсальную кодировку - Unicode Consortium

Символы также нумеруются, на данный момент закодировано 1 миллиона символов

(можно поместить в три байта)

У этой кодировки есть структура :

1. *basic multilingual plane – основной набор символов, туда входят самый популярные символы языков. Имеют номера < 2^16 (входят в первые два байта), причем первые 128 номеров совпадают с ASCII*
2. *внутри basic plane также структурирован, символы каждого национального алфавита размещены в соответствующую группу, причем в группах оставлены неиспользованные символы (запас)*
3. *символы с кодом > 2^16 : вымершие алфавиты, смайлики*

Также в unicode есть возможность представлять символ как объединение двух или более. Два символа записываются рядом, программа которая поддерживает объединение, отображает их как один.

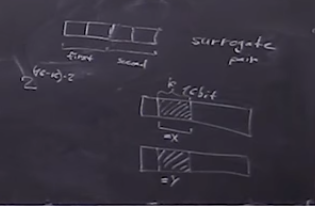
Особенность unicode – нумерация единая, однако кодировок все же несколько.

1. Самая простая кодировка ucs-2, ucs-4. Число означает сколько байт занимает один символ или code point (правильный термин для символов в таблице соответствия unicode).

Usc2 – слишком маленький размер, его не хватило, ucs-4 слишком боьшой.

1. Utf-16 кодировка похожая на ucs-2, но позволяет кодировать все нужные символы.

16-количество бит. Идея кодировки : для BMP используется та же кодировка что в ucs-2, а для всех остальных предусмотрены специальные зарезервированные символы(сурогатные code points), через них выражают символы с кодом >2^16.



Достоинства UTF-16

* Можем скормить текст как программам которые понимают только ucs-2 так и программам кот уже понимают utf-16
* Можем начать читать текст с любого байта кратного двум.
* 1) попали в начало сурагатной пары – спокойно считываем
* 2) попали в середину (на второе слово в сурагатной пары) – идем либо назад либо пропускаем и идем вперед
* \*так мы не будем показывать фигню всякую\*

Недостатки UTF-16 & UCS-2

* Нет возможности читать с произвольного байта. В целом проблему можно решить, используя эвристические алгоритмы. Либо же можно искать пробелы
* Усложняется декодирование (для utf-16)
* Не самая оптимальная по занимаемому месту (если текст состоит только из ascii символов, например)

1. UTF-32 (4 BYTE) по идее совпадает с ucs-4
2. UTF-8 самая оптимальная кодировка

Задача : если текст состоит только из ascii символов, то закодированный текст должен выглядеть также как в кодировке ascii

Как работает кодировка – см лекции

Плюсы:

- можем начать декодировать с любого места

- экономия памяти

Минусы:

- Еще сложнее декодер

- не можем примерно понять сколько примерно даже символов у нас, придется просматривать весь текст

- не можем производить индексирование (например k-ый символ найти)