Семинар 6: Указатели и кодировки

10 Декабря, 2019

Указатели: освежаем в памяти

- Указатель — номер байта в адресном пространстве
- Любой указатель имеет тип (например, int$*$, long$*$, ...)
- Разыменовывание (dereference) — обращение к памяти, на которую указывает указатель
- Разыменовывание NULL-указателя — undefined behavior!
- void$*$ ссылается на что угодно, нельзя разыменовать
- Приведение указателей (casting): void$*$ v = ...; int$*$ a = v;

Выравнивание

- Говорят, что указатель *выравнен (aligned)*, если адрес, на который он ссылается, кратен K байтам
- K называют *выравниванием указателя (alignment)*
- Или говорят, что *указатель выровнен по границе K байт*
- В современных процессорах обычно используются выравнивания вида $K = 2^N$

Выравнивание: зачем?

- Некоторые процессоры читают память по выровненным адресам быстрее
- Некоторые процессоры читают память *только* по выровненным адресам
- Если пишите переносимый (portable) код, то лучше всё выравнивать как надо!

Выравнивание скалярных типов в C

- char выравнивается по границе 1-ого байта (иногда говорят, что у него нет выравнивания)
- short выравнивается по границе 2-ух байт
- int и float выравниваются по границе 4-ёх байт
- Выравнивание остальных типов зависит от битности процессора
- x86: long, long long и double выравниваются по границе 4-ёх байт
- x86-64: long, long long и double выравниваются по границе 8-ми байт

Выравнивание скалярных типов в C

- Массивы, выделенные на стеке имеют выравнивание типа элемента
- malloc гарантирует, что в выделенная память выровненна так, что в ней можно разместить любой тип

Выравнивание структур в C

- Члены структур располагаются рядом
- Но если им не хватает выравнивания, компилятор «добивает» структуру pad'ами
- Выравнивание структуры — максимальное выравнивание среди всех выравниваний её членов

Выравнивание: offsetof

```
#define offsetof(st, m) ((size_t)&(((st *)0)->m))
```

Кодировки

- character — что-то, что мы хотим представить
- character set — какое-то множество символов
- coded character set (CCS) — отображение символов в уникальные номера
- code point — уникальный номер какого-то символа

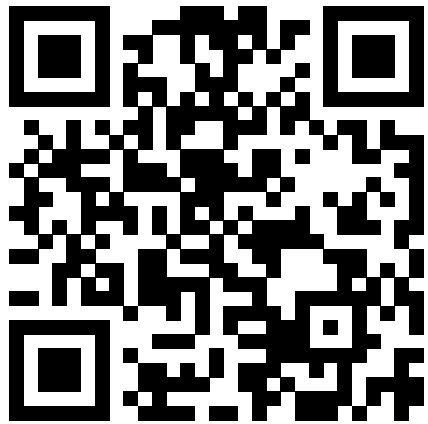
ASCII

- American Standard Code for Information Interchange, 1963 год
- 7-ми битная кодировка, то есть кодирует 128 различных символов
- Control characters: с 0 по 31 включительно, непечатные символы, мета-информация для терминалов

Unicode

- Unicode is a computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems.
- Code point'ы обозначаются как U+<число>
- Например: $\times$ = U+00D7, a = U+0061
- Определяет CCS, но не саму кодировку!

Unicode Charts



UTF-8

- Unicode Transformation Format
- Определяет способ как будут преобразовываться code point'ы
- Переменная длина: ASCII символы кодируются одним байтом, кириллица — 2-умя

UTF-8: способ представления

Диапазон	Биты	Байт 1	Байт 2	Байт 3	Байт 4
0000-007F	7	0xxxxxxx	–	–	–
0080-07FF	11	110xxxxx	10xxxxxx	–	–
0800-FFFF	16	1110xxxx	10xxxxxx	10xxxxxx	–
10000-10FFFF	22	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

UTF-8: overlong encoding

- Какой code point кодирует последовательность 00100000?

UTF-8: overlong encoding

- Какой code point кодирует последовательность 00100000?
  U+0020

UTF-8: overlong encoding

- Какой code point кодирует последовательность 00100000?
  U+0020
- А вот эта: 11000000 10100000?

UTF-8: overlong encoding

- Какой code point кодирует последовательность 00100000? U+0020
- А вот эта: 11000000 10100000? Тоже U+0020!

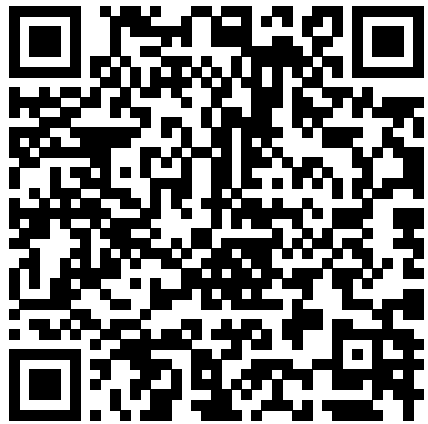
UTF-8: overlong encoding

- Какой code point кодирует последовательность 00100000? U+0020
- А вот эта: 11000000 10100000? Тоже U+0020!
- Это называется overlong form или overlong encoding
- С точки зрения стандарта является некорректным представлением

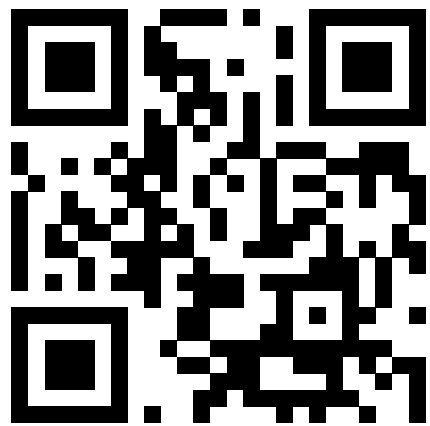
UTF-16

- Использует 16 бит (2 байта) для кодировки
- Кодируют 0000-D7FF и E000-FFFF обычными short'ами
- Суррогатная пара для кодировки больших code point'ов: 110110xxxxxxxxxx 110111xxxxxxxxxx

UTF-16: подводные камни



UTF-8 everywhere!



UTF-32

- Использует 32 бита (4 байта) для кодировки
- Используется во внутреннем представлении строк в некоторых языках (например, Python)
- Позволяет обращаться к произвольному code point'у строки за O(1)

wchar.h

- Заголовочный файл для работы с wide strings
- Типы: wchar_t (вместо char) и wint_t (вместо char + EOF)
- I/O-операции: wprintf, wscanf, getwc, getws, ...
- Utility functions: wcscmp, wcslen, wcsstr, ...
- Для одного символа: iswalnum, iswalpha, iswdigit, ...

Danke!