CS 107
Lecture 4: C-strings

Friday, October 6, 2017

Computer Systems
Fall 2017
Stanford University
Computer Science Department

Lecturers: Julie Zelenski and Chris Gregg
Check-in
Today’s topics

- Everything you wanted to know about C-strings (but were afraid to ask)
- Char/ascii
- String constants, string as “abstraction” (albeit a leaky one)
- C library functions <string.h>
- Under the hood: C-string as pointer/array of char
- Array indexing vs pointer arithmetic
- Ask questions today (and every day!)
Character data

- **char is 1 byte**
  (by definition in standard, never need to compute sizeof(char)

- **May be signed or unsigned by default**
  Of consequence when promoting to larger type but not much else

- **Standard ASCII maps entries 0x00 - 0x7f to letters, digits, punct**
  8th bit used for parity check, error detection

- **No consensus for character map of extended range**

- **Unicode**
  more robust/flexible but more complex, larger storage needs

- **What can you do with a char?**
Let's code!

/afs/ir/class/cs107/samples/lect4

chars.c
Representing a sequence of chars

How to represent sequence of chars?
- Stuff into a 4-byte or 8-byte word?
- Array/vector/list of char?
- Something else?

C language features generally map straight to machine structures
- We saw this with behavior/operations of integers
- How does system support sequence of anything?

C-string is a pointer!
- char *str = "binky";

What can you do with a C-string?
Let’s code!

/afs/ir/class/cs107/samples/lect4

client.c
Segmentation fault

**Address space**
- Indexed bytes
- Start address to end
- 64-bit address space theoretically address 16EiB
- Hardware limit to fewer

**Memory divided into segments**
- Code, data, stack, heap each have their own segment

**Attempt to access invalid address -> segmentation fault**
- Attempt to write to address in read-only segment OR
- Read/write address past segment boundary OR
- Read/write address that is not within valid segment