C Review

Bryce Boe

2013/10/02

CS24, Fall 2013

Outline

- Review Lab 1
- Research consent forms
- C Review

LAB 1 REVIEW

Lab 1 Common Problems

- Performing *failure* testing too late
- Not handling the 0 case
- Whitespace issues
- Others?

Lab 1 Solution

- <In-class review of lab 1 solution source code>

CS16 REVIEW

What are the sizes of the following primitive types (x86– Intel 32 bit)?

- int
- float
- double
- char
- void*
- int*
- char*

Arrays

- Reference cs16_review_arrays.c
- Unitialized:
  - int foo[16];
  - char bar[1024];
- Fully initialized
  - int blah[] = {1, 2, 3, 0xDEADBEEF, 0b1010};
  - char msg[] = “hello world”;
  - char other[] = {‘a’, ‘b’, 99, ‘d’, ‘e’};

Structures

- Structures (struct keyword) allows you to define your own types (see cs16_review_struct.c)

```
struct Point {
  int x;
  int y;
  char *name;
};

struct Point p1;  // Uninitialized
struct Point p2  = {16, 32, “some point”};  // Initialized
```

Pointers

- A primitive type that stores the address to where the *data* is actually stored in memory
- When accessing elements of a **struct**, use `->` to automatically dereference the object

```
struct Point *p1 = malloc(sizeof(struct Point));
p1->name = “some name”;
(*p1).name = “some name”;  // the same as above
free(p1);  // Always free the memory when done
```

C-strings

- C-strings are an array of characters followed by ‘\0’ (0b0000)
- char local_string[] = “hello world”;
- char manual_string[] = {‘a’, ‘b’, ‘c’, ‘\0’};
- char not_a_cstring[] = {‘x’, ‘y’, ‘z’};
- char *pointer_string = “hello world”;