More C

CSE 413, Autumn 2007

(A few examples borrowed from cse303)

Topics

- Scope
- File layout
- Function declarations
- Parameter passing
- structs

Storage, lifetime, and scope

- **Global variables** allocated before main, deallocated after main.
  - » Scope is entire program.
  - » Usually bad style, similar to public static Java fields.
- **Static global variables** like global variables but
  - » Scope is just that file, kind of like private static Java fields.
  - » Related: static functions cannot be called from other files.
- **Static local variables** like global variables (!) but
  - » Scope is just that function, rarely used.
- **Local variables** allocated "when reached", deallocated "after that block"
  - » Scope is that block.
  - » Like local variables in Java.

File Layout Style

```
// includes for functions, types defined elsewhere (just
   prototypes)
#include <stdio.h>
#include ...

// global variables (usually avoid them)
int some_global;
// static variables (use only in this file)
static int this_file_arr[7] = { 0, 2, 4, 5, 9, -4, 6 };

// function prototypes for forward-references (to get around
// uses-follow-definition rule)
void some_later_fun(char, int); // argument names optional

// function definitions
void f() { ... }
void some_later_fun(char x, int y) {...}

int main(int argc, char**argv) {...}
```


Function Declarations

- A function must be defined or declared before it is used.
  - » return type assumed int, ... complains when sees actual definition if it has return type other than int
- Linker error if something is used but not defined.
- Use -c to compile without linking (more later).
- To write mutually recursive functions, you just need put a declaration before the definition.

C "Quirks"

- Declarations only at the beginning of a "block"
  - » e.g. the beginning of a function
  - » Just put in braces if needed to create a new block
- No built-in boolean type; use ints (or pointers)
  - » Anything but 0 (or NULL) is true.
  - » 0 and NULL are false.

Declaration Gotchas

- You can put multiple declarations on one line, e.g.,

```
int x, y;        or
int x=0, y;      or
int x, y=0;
```

- But watch out....

`int *x, y;` means: `int *x; int y;`

(you usually are trying to say: `int *x, *y;`)

7

Function Arguments

- Storage and scope of arguments is like for local variables.
- But initialized by the caller ("copying" the value)
- So assigning to an argument has no affect on the caller.
- But assigning to the space pointed-to by an argument might.
- (see function call example)

8

What happens when we call h?

```
int* f(int x) {
  int *p;
  if(x) {
      int y = 3;
      p = &y;
      }
  y = 4;
  *p = 7;
  return p;
}
void g(int *p){
  *p = 123;
}

void h() {
  g(f(7));
}
```

9

Structs

- A struct is a record. (similar to a Java object with no methods.)
  - » x.f is for field access.
  - » (*x).f in C is like x.f in Java.
  - » x->f is an abbreviation for (*x).f.
- There is a huge difference between passing a struct and passing a pointer to a struct.
- (see struct example code)

10