C for Java Programmers & Lab 0

Don Porter

Portions courtesy Kevin Jeffay

Same Basic Syntax

- Data Types: int, char
  - void - (untyped pointer)
  - Can create other data types using typedef
- No Strings - only char arrays
  - Last character needs to be a 0
    - Not '\0', but '\0'

struct – C’s object

- typedef struct foo {
  int a;
  void *b;
  void (*op)(int c); // function pointer
} foo_t;  // <------type declaration
- Actual contiguous memory
- Includes data and function pointers

Pointers

- Memory placement explicit
  (heap vs. stack)
- Two syntaxes (dot, arrow)
  int main {
    struct foo f;
    struct foo *fp = &f;
    f.a = 32; // dot: access object directly
    fp->a = 33; // arrow: follow a pointer
    fp = malloc(sizeof(struct foo));
    fp->a = 34;
    ...
  }

Function pointer example

- void myOp(int c){/*...*/}
  /...*/
- struct foo *myFoo = malloc(sizeof(foo_t));
- myFoo->op = myOp; // set pointer
  /...*/
- myFoo->op(5); // Actually calls myop

More on Function Pointers

- C allows function pointers to be used as members of a struct or passed as arguments to a function
- Continuing the previous example:
  void myOp(int c){/*...*/}
  /...*/
  foo_t *myFoo = malloc(sizeof(foo_t));
  myFoo->op = myOp; // set pointer
  /...*/
  myFoo->op(5); // Actually calls myop
No Constructors or Destructors

- Must manually allocate and free memory - No Garbage Collection!
  - void *x = malloc(sizeof(foo_t));
    - sizeof gives you the number of bytes in a foo_t - DO NOT COUNT THEM YOURSELF!
  - free(x);
    - Memory allocator remembers the size of malloc’ed memory
- Must also manually initialize data
  - Custom function
  - memset(x, 0, sizeof(*x)) will zero it

Memory References

- ‘.’ - access a member of a struct
  - myFoo.a = 5;
- ‘&’ - get a pointer to a variable
  - foo_t *fPointer = &myFoo;
- ‘->’ - access a member of a struct, via a pointer to the struct
  - fPointer->a = 6;
- ‘*’ - dereference a pointer
  - if(5 == *intPointer)[...]
    - Without the *, you would be comparing 5 to the address of the int, not its value.

Int example

```c
int x = 5; // x is on the stack
int *xp = &x;
*xp = 6;
printf("%d\n", *xp); // prints 6
xp = (int *)0;
*xp = 7; // segmentation fault
```

Memory References, cont.

- ‘[]’ - refer to a member of an array
  - char *str = malloc(5 * sizeof(char));
  - str[0] = ‘a’;
  - Note: *str = ‘a’ is equivalent
  - str++; increments the pointer such that *str == str[1]

The Chicken or The Egg?

- Many C functions (printf, malloc, etc) are implemented in libraries
- These libraries use system calls
- System calls provided by kernel
- Thus, kernel has to "reimplement" basic C libraries
  - In some cases, such as malloc, can’t use these language features until memory management is implemented

For more help

- man pages are your friend!
  - [not a dating service]!
  - Ex: ‘man malloc’, or ‘man 3 printf’
    - Section 3 is usually where libraries live - there is a command-line utility printf as well
- Use ‘apropos term’ to search for man entries about term
- The C Programming Language by Brian Kernighan and Dennis Ritchie is a great reference.
Lab 0 Overview

- C programming on Linux refresher

Lab 0 - Overview

- Write a simple C character stream processing program on Linux
- Read in characters from “standard input,” write 80 character lines to “standard output” replacing:
  - Every enter/return character (newline) by a space
  - Every adjacent pair of percents “%%” with an “*”

- Example (for a 30 character output line):
  - The string...
  - ...is output as:

◆ This is the only output your program should generate
  - There should be no prompts, debugging messages, status messages, ...
  - Note that your output will be interleaved with your input on the console (indicated in purple above)
  - This is fine!
  - (You can eliminate this if you use "I/O redirection")

Submitting Homework Assignments

- You should all have Linux accounts in the Department
  - If you don’t, go to the let me know ASAP!
  - If you need to have your password reset visit https://www.cs.unc.edu/webpass/onyen/
- Create the directory structure comp530/submissions in your Linux home directory
- Execute the magic incantations:
  - `fs sa ~/comp530/submissions system:anyuser none`
  - `fs sa ~/comp530/submissions porter read`
  - `fs sa ~/comp530/submissions sytang read`
  - `fs sa ~/comp530/submissions rohan read`

Execute these instructions before the next steps!

Submitting homework

- For each assignment in this course, create a subdirectory named HWx in comp530/submissions
  - Keep all files required to execute your program in this subdirectory
- For example, for lab0, create the subdirectory “lab0” in ~/comp530/submissions
- For lab0 name your program warmup.c
  - Note that Linux names are case sensitive so case matters!

If you don’t follow these instructions exactly, your HW will not be graded!
Submitting Homework

- Send email to comp530ta-f16@cs.unc.edu when your program is ready to be graded
  - Include your Linux login id in your email so we know where to find your files
  - If you don’t send email your program will never be graded!
  - If you’re late with an assignment simply send email when the program is ready for grading
  - Whether or not a program is on-time or late will be determined solely by the latest modification time of the files in the HWx subdirectory
- After the due date do not edit/modify any files in the HWx subdirectory
  - If you need to reuse files for the next assignment, or any other purpose, copy the required files to a new submissions subdirectory

Lab 0 Programming Notes

- The machines you should use for programming are:
  - classroom.cs.unc.edu (primary)
  - snapper.cs.unc.edu (secondary)
- Access either machine via a secure shell (secure telnet) application on your PC
- You can develop your code anywhere you like but…
- Your programs will be tested on classroom and correctness will be assessed based on their performance on classroom
  - Always make sure your program works on classroom!

Grading

- Programs should be neatly formatted (i.e., easy to read) and well documented
- In general, 75% of your grade for a program will be for correctness, 25% for programming style
  - For this assignment, correctness & style will each count for 50% of your grade
- Style refers to…
  - Appropriate use of language features, including variable/procedure names, and
  - Documentation (descriptions of functions, general comments, use of invariants, pre- and post conditions where appropriate)
  - Simple test: Can I understand what you’ve done in 3 minutes?
- Correctness will be assessed comprehensively!
  - You’ve got to learn to test for “edge” and “corner cases”

Dr. Jeffay’s Experience

- Programs that “mostly work” don’t cut it in a senior-level course!

Honor Code: Acceptable and Unacceptable Collaboration

- Working in pairs on programming assignments is OK
  - But you can only collaborate with other students in the course
  - Every line of code handed in must be written exclusively by team members themselves, and
  - All collaborators must be acknowledged in writing
- Use of the Internet
  - Using code from the Internet in any form is not allowed
  - Websites may be consulted for reference (e.g., to learn how a system call works)
  - But all such websites used or relied on must be listed as a reference in a header comment in your program
  - Warning: Sample code found on the Internet rarely helps the student