

# CSE 220 – Systems-Level Programming

Hussein Badr

Department of Computer Science  
Stony Brook University  
Stony Brook, NY 11794-4400 USA  
(631) 632-8455

[badr@cs.stonybrook.edu](mailto:badr@cs.stonybrook.edu)

# Brief Course Description

Introduces systems-level programming concepts using the C language and assembly language, and explores the relation of respective programs in these languages.

## Topics include:

- internal data representation
- basic instructions and control structures
- arithmetic operations
- pointers
- function calls and parameter passing
- memory allocation
- logical and shift operations
- linking and loading

# Getting In to the Class

- **This class:** fill out questionnaire and hand back.  
**Note:** ALL students – both registered and not registered – must fill out the questionnaire.
- It will almost certainly take me into next week to chase down students registered but not showing up to class and get them to drop the course or deregister them in order to make room for those of you showing up and wanting to register.
- You *must* have taken either CSE 114 or CSE 160 (or passed proficiency) and received grade of 'C' or better.
- You *must* be officially admitted to the CSE major.

Instructional resource limitation issues have forced us to limit enrollment to CSE majors.

# Course Information

- All info on Web:

`http://www.cs.stonybrook.edu/~cse220`

Please check the course page frequently.

# Official Course Objectives

- Provide a systematic introduction to the C language for students who already have acquired a solid understanding of object-oriented programming.
- Introduce the basics of assembly language programming and explore the relation to C programming.
- Develop an understanding of key aspects of efficient execution of high-level language programs.

# Outline of Course

- Background topics (*approx. 2 weeks*)
- Introduction to C programming (*approx. 6 weeks*)
- Introduction to Assembly Language programming (*approx. 6 weeks*)

# Textbooks

- *The C Programming Language (2nd Edition)* [Paperback], Kernighan and Ritchie, Prentice Hall, 1988.
  - ISBN-10: 0131103628
  - ISBN-13: 978-0131103627

This is the classic reference on the C programming language, written by the language designers.

- *MIPS Assembly Language Programming* [Paperback], Britton, Prentice Hall, 2004.
  - ISBN-10: 0131420445
  - ISBN-13: 978-0131420441

# Course Components

- **Lectures:** Presentation of the main topics.
- **Recitations:** Additional examples, discuss homework, and maybe also short quizzes every now and then.
- **Textbooks:** Serve as detailed references.
- **Homework:** Approximately weekly.
- **Exams:** Two in-class midterms and a final exam.

# Grading

- **Recitation** (10%) (attendance/quizzes?)
- **Homework** (20%)
- **Midterm Exam #1** (20%)
- **Midterm Exam #2** (20%)
- **Final Exam** (30%)

# Official Course Software Environment

The course presupposes the following software environment:

- *GNU C Compiler (gcc) and GNU Debugger (gdb)*
  - Comes with Linux/Unix systems
  - Available for Macintosh systems.
- *Mars MIPS Simulator*
  - Distributed as an runnable Java Archive (“jar”).
  - Requires a Java Runtime (JRE) installation.

- *Editor/Environment for editing C programs*
  - Your choice, but helpful if it understands something about C syntax.
  - Eclipse *C Development Toolkit* (CDT) looks promising.

You can use other tools/IDEs if you like, but what you submit must *not* contain generated code and it must compile and run under a “vanilla” ANSI C compiler. More specifically, it must compile and run using the gcc compiler on Sparky, on which you will be electronically handing in your programming assignments.

One purpose of the course is to learn what happens “under the covers.” Even if you use an IDE to do your C programming, you will still be responsible for knowing how to compile and run using command-line tools (on Sparky, for example). This is part of skill set you need to develop on your way to becoming Computer Science professionals.

# Academic Dishonesty

Everyone will submit individual homeworks in this course.

- You may *discuss* course material with others.
- Your *wroteups* and *coding* must be *your own*.
- It goes without saying that viewing or copying others' work during an exam is *not* permitted.
- *Any* evidence of shared wroteups or code, or copying during an exam will result in charges being submitted to the CEAS CASA Committee.

# Academic Dishonesty: Why Do I Care?

Individual effort is very important in learning to program.

- *Doing the HW (all of it!) is the main way you will learn.*

There is only one reason to turn in work not your own: to try to get a better grade without doing all the required work.

- This is the definition of “Academic Dishonesty.”

When people graduate without learning the material, it degrades the good reputation and value of a SBU degree for *everyone*.