

Introduction



CS 351: Systems Programming
Michael Lee <lee@iit.edu>

Michael Lee

- lee@iit.edu
- <http://moss.cs.iit.edu>
- Office hours: Wed/Fri 1-3PM
 - By appointment only! (Zoom/In-person)

Agenda

- Syllabus & Administrivia
- Course overview (“Systems Programming”)

§Syllabus

Prerequisites

- “substantial” programming experience
- data structures: concepts & implementation
- basic run-time analysis (big O)
- knowledge of (any) assembly language
- computer organization essentials

- computer organization essentials:
 - data representation (binary, two's comp, f.p. inaccuracy, etc.)
 - von Neumann model
 - CPU, memory, I/O
 - stack usage / conventions

Online resources

1. Course website

moss.cs.iit.edu/cs351

- static information
 - lecture calendar, assignment writeups, slides, screencasts, links, etc.

The screenshot shows a web browser window for the course website. The title bar says "CS 351: Systems Programming". The main content area has two sections: "Announcements" and "Calendar".
Announcements:

- The Zoom meeting ID for our online lectures is 893 7228 5322 -- you can join using [this link](#).
- Lecture recordings can be found in the [class Panopto folder](#).
- We will be using Discord to run labs, and it will also serve as our peer support and Q/A forum. If you aren't already a member, please join at <https://discord.gg/exbFK5b>

Calendar:

Please note that readings for a given lecture should ideally be reviewed before coming to class, and will likely need to be revisited afterwards. Most readings are from "Computer Systems: A Programmer's Perspective, 3rd Edition" (CSAPP3e). Lecture slides/notes will be updated after class, when available.

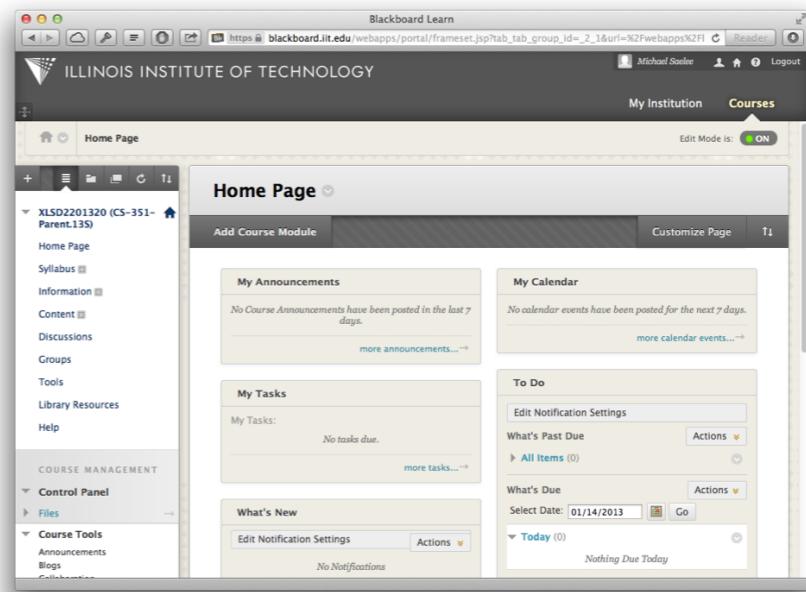
The lecture calendar is tentative and may be updated.

Dates	Topic	Notes	Reading(s)
May 25	Syllabus and Course overview	01-prelim	Syllabus
May 25 - May 27	C Primer	02-c_x86-64-guide	K&R, Pro Git chapters 1, 2
Jun 01	The Process	03-lpprocess	CSAPP 8.1-8.2
Jun 01	Process Management: fork/exec	04-procmgmt-1	CSAPP 8.3-8.4.2
Jun 03	Process Management: wait/exec	05-procmgmt-2	CSAPP 8.4.3-8.4.6
Jun 03 - Jun 08	Process Management / Unix Shell	06-procmgmt-3	CSAPP 8.5-8.6
Jun 10	Memory Hierarchy and Caching	07-caching	CSAPP 6.1-6.6
Jun 15	Cache Optimization		
Jun 16	Midterm Exam		

Online resources

2. Blackboard

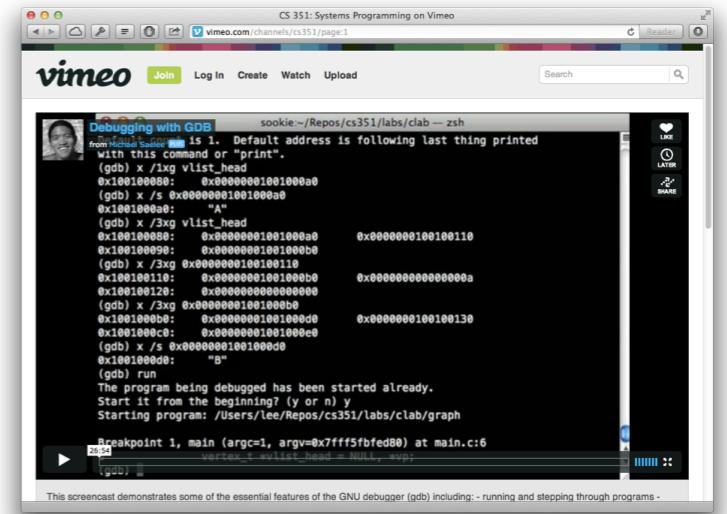
- grade spreadsheet
- online exams



Online resources

3. Vimeo channel: screencasts

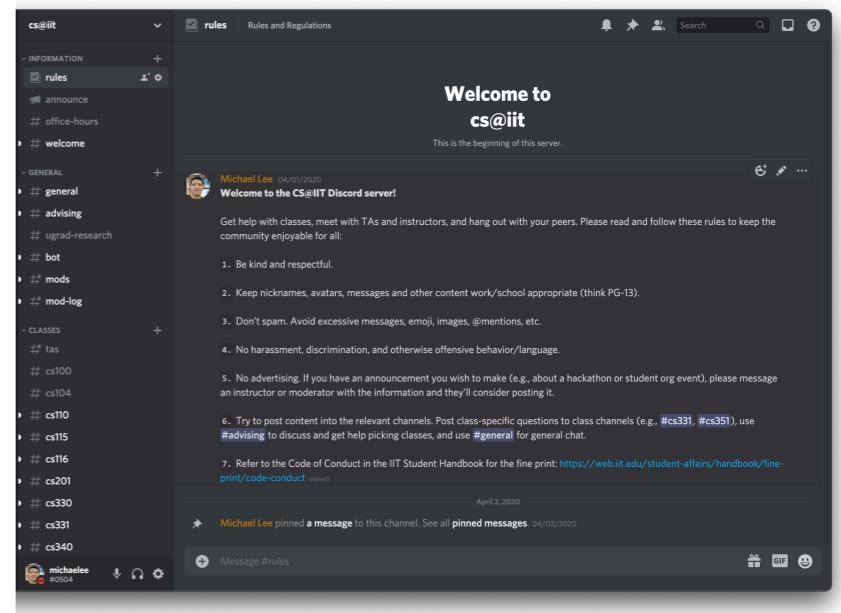
- vimeo.com/channels/cs351
- walkthroughs & tutorials
(check before starting labs!)



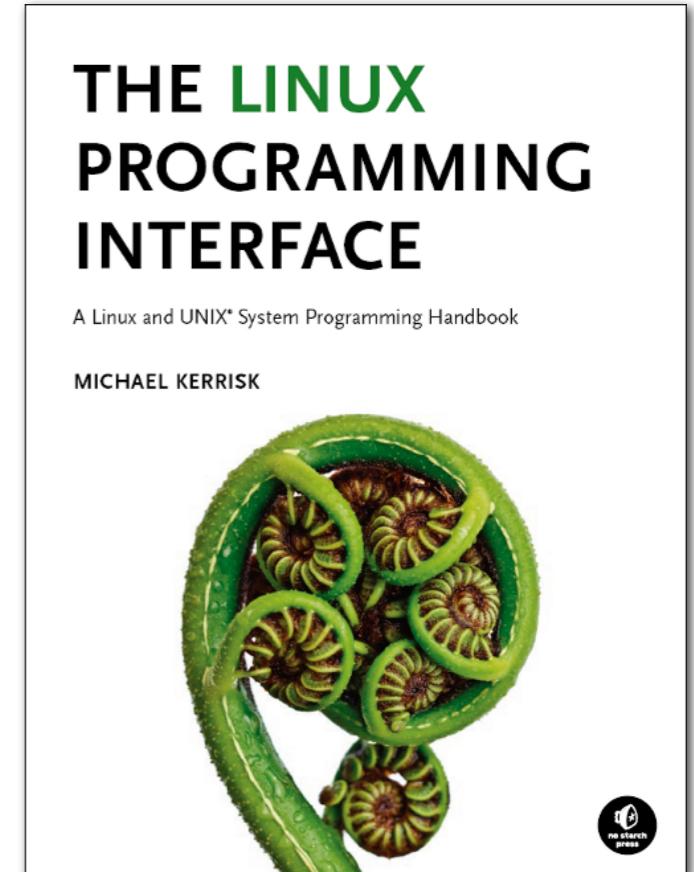
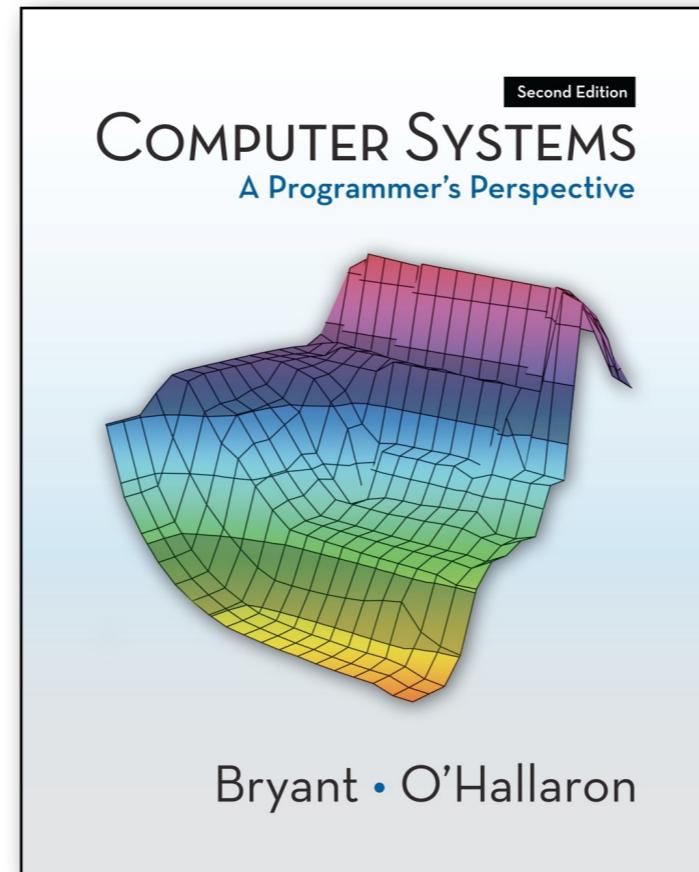
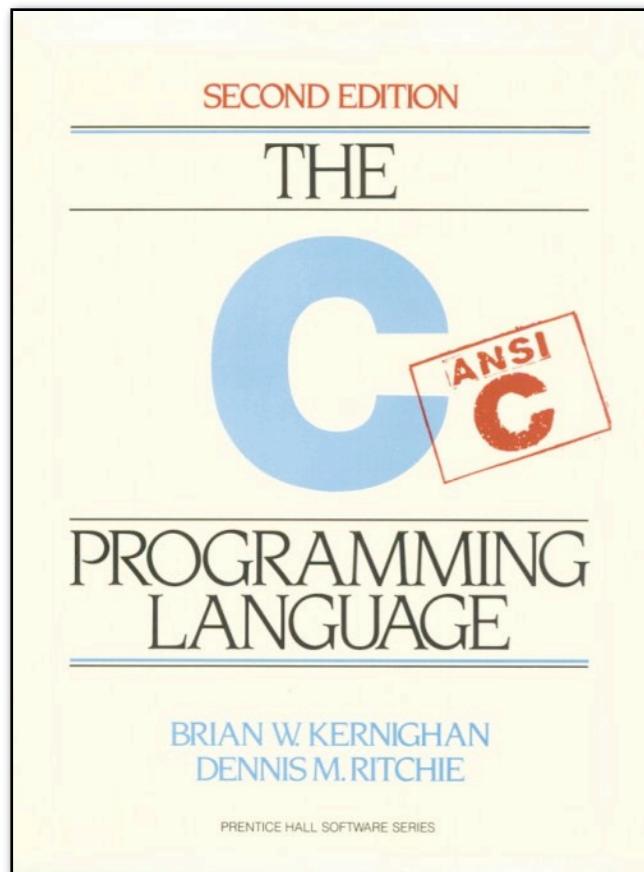
Online resources

4. Discord: discussion forum

- peer support / discussion
- text/audio chat
 - + screensharing



Textbooks



Grading

- 50% Labs
- 25% Midterm exam
- 25% Final exam

Grade Scale

```
char letter_grade(float score) {  
    if (score >= 90.0)          return 'A';  
    else if (score >= 80.0)    return 'B';  
    else if (score >= 70.0)    return 'C';  
    else if (score >= 60.0)    return 'D';  
    else return 'E';  
}
```

Labs

- 5-7 fairly substantial machine problems
- real-world application of concepts covered in lecture & textbook
- late policy: 7 day “late pool”
 - 1 day at a time, at your discretion

Exams

- Midterm exam on March 3rd
- Final exam is nominally cumulative
- Scores may be linearly scaled so that median/mean (whichever lower) is 70%

§ Course Overview

“Systems Programming”

system |'sistəm|

noun

1 a set of connected things or parts forming a complex whole

(New Oxford American Dictionary)

“Systems Programming”

- Programming the *operating system*
- What does *that* mean?

OS vs. OS *kernel*

- OS kernel \approx smallest subset of OS code needed to bootstrap system and provide basic services to user programs
- “smallest” is debatable

How to “program” it?

- Require some API
 - Application Programming Interface
 - A collection of (documented) functions
 - e.g., get/put/del for a hashtable

OS API

- a.k.a. “system call” interface
 - OS as a very low-level library
- common purpose: provide services to user level programs
 - *def:* program in execution = ***process***

The Process

- A program in execution
- Code + Data { global, local, dynamic }
- + OS kernel data
- OS hides complexity of machine from processes by creating *abstractions*

AN x64 PROCESSOR IS SCREAMING ALONG AT BILLIONS OF CYCLES PER SECOND TO RUN THE XNU KERNEL, WHICH IS FRANTICALLY WORKING THROUGH ALL THE POSIX-SPECIFIED ABSTRACTION TO CREATE THE DARWIN SYSTEM UNDERLYING OS X, WHICH IN TURN IS STRAINING ITSELF TO RUN FIREFOX AND ITS GECKO RENDERER, WHICH CREATES A FLASH OBJECT WHICH RENDERS DOZENS OF VIDEO FRAMES EVERY SECOND

BECAUSE I WANTED TO SEE A CAT JUMP INTO A BOX AND FALL OVER.



“Abstraction”

<http://xkcd.com/>

Primary Abstractions

- Logical control flow
- Exceptional (extra-process) control flow
- Logical address space
- “I/O” (via uniform APIs)
- Interprocess Communication

In the old days . . .

- ... every program had to include its own implementation of all the above!
- Now, OS simplifies life for all of us.
 - Only need to know how to *use* them, not how they're *implemented*.

But!

- In this class we dig a bit deeper
 - What facilities are encapsulated by syscalls?
 - What limitations/restrictions do they have?
 - Why are they designed the way they are?
 - How do they work behind the scenes?

But why should I care?

- *efficiency*: know how to use tools optimally; reuse existing features and design/layer new ones appropriately
- *robustness*: avoid bugs/failures & know how to diagnose and fix them

the real reason: it's fun to take things apart!





goal: turn you into a **hacker**

(or: make you a **better** hacker)

hacker |'hakər|

noun

- 1 A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary.

The Jargon File, version 4.4.7

Our tools (& approach)

- C & Linux
 - C: low-level language
 - GNU Linux: open source kernel & tools
 - GNU gdb & gcc; debugger & compiler

Fourier

- All labs must be tested and submitted on the class Linux server: fourier.cs.iit.edu
- You should have received an e-mail with account info already
- If off-campus, must connect via IIT VPN
- Log in via SSH client

Git & GitHub

- All labs are distributed using Git via GitHub
 - Distributed VCS + platform
- Typical workflow: (1) accept invitation and get a private copy of the assignment repo, (2) clone repo on fourier, (3) work on assignment, (4) submit via commit & push

No lab sessions next week!

- MLK day
- But! I will be releasing lab 0 — no coding, just familiarizing you with lab logistics