Introduction to Computer Systems

COMP 321

Dave Johnson Alan Cox Scott Rixner

1

Course Goals

A programmer's view of how programs execute on a computer system

- Building on the hardware's execution
- ***Using services provided by the operating system and interacting with the OS***

Make you a better programmer

- Better understand program behavior and thus the likely causes of bugs
- Write more reliable, more correct programs

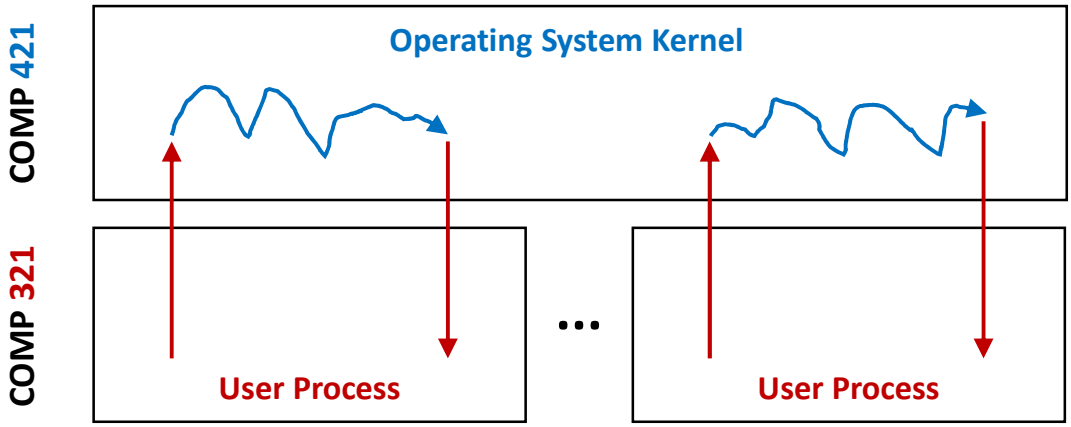
Prepare for later systems courses

- Operating systems (COMP 421), distributed programs (COMP 413), security (COMP 427), networking (COMP 429), compilers (COMP 412), ...

2

COMP 321 vs. COMP 421

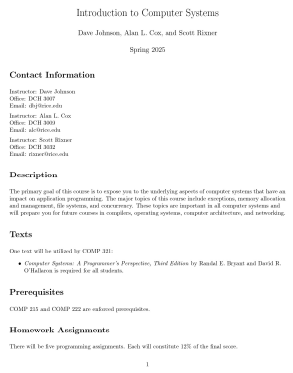
User programs making typical "kernel calls":



Syllabus Overview

This is only an overview of the syllabus

- Please read the ***full*** syllabus, available on Canvas, course website, and in Esther
- I will talk here only about some parts of the syllabus
- You are responsible for ***everything*** in the full course syllabus



Introduction to Computer Systems

Dave Johnson, Alan L. Cox, and Scott Rixner

Spring 2025

Contact Information

Instructor: Dave Johnson
Office: DCH 3007
Email: dbj@rice.edu

Instructor: Alan L. Cox
Office: DCH 3093
Email: alc@rice.edu

Instructor: Scott Rixner
Office: DCH 3032
Email: rixner@rice.edu

Description

The primary goal of this course is to expose you to the underlying aspects of computer systems that have an impact on application programming. The major topics of this course include exceptions, memory allocation and management, file systems, and concurrency. These topics are important in all computer systems and will prepare you for future courses in compilers, operating systems, computer architecture, and networking.

Texts

One text will be utilized by COMP 321:

- *Computer Systems: A Programmer's Perspective, Third Edition* by Randal E. Bryant and David R. O'Hallaron is required for all students.

Prerequisites

COMP 215 and COMP 222 are enforced prerequisites.

Homework Assignments

There will be five programming assignments. Each will constitute 12% of the final score.

1

Prerequisites

Prerequisite courses and prerequisite understanding

- COMP 215
  - Basic understanding of programming and ability to program
- COMP 222
  - Topics including machine representation of data, basic assembly language understanding, linking, basic memory hierarchy, processor architecture
  - ***Knowledge of and ability to program in the C programming language***

Grading

5 homework programming assignments

- Each worth 12% of the semester grade

Midterm exam (closed book)

- 16% of the semester grade
- ***In person***, on ***Wednesday, February 26***, 7:00-9:00 PM
- If you have a legitimate scheduling conflict with this time, see the syllabus

Final exam (closed book)

- 24% of the semester grade
- ***In person***, at a date and time ***to be scheduled by the Registrar***

Weekly Ungraded Practice Labs

The lab sections each meet once a week

- Hands on practice with material from the lectures
- Introduces new material beyond the lectures
- Programming exercises and experience
- General programming tips
- Very helpful towards the graded homework programming assignments

Small extra credit grade "bump" for attending the labs

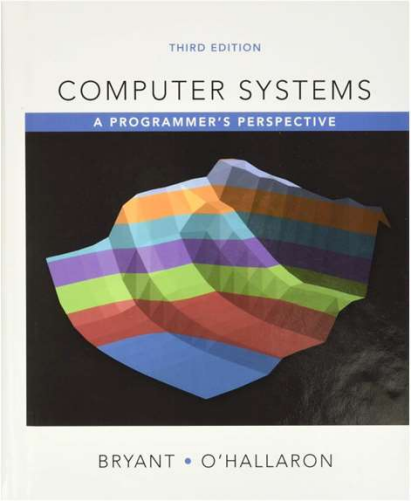
- Must miss attending no more than 3 days of the lab meetings
- Bump is enough to move you "above the line" (e.g., from B+ to A-) if your grade is already close to the line

Textbook

Computer Systems: A Programmer's Perspective, Third Edition, 2016

- Bryant and O'Hallaron
- You should already have a copy of this from COMP 222
- We will also cover some things that are not in this book

Also recommended: Head First C, Griffiths and Griffiths, O'Reilly Media, 2012



Course Communication

COMP 321 course website:

http://www.clear.rice.edu/comp321

- Schedule of lectures, labs, homeworks, and exams
- Copies of lecture slides and labs
- Contact information for the course staff
- Schedule of office hours

COMP 321 Canvas site

- Recording grades
- Time-critical announcements

Course Communication

Piazza web-based question and answer platform

- Get your questions answered by TAs, instructor, and fellow students
- Read ***all*** questions and ***all*** answers, and check for duplicates before posting
- Be careful to not inadvertently violate the Honor Code in what you post
  - If you are asking about the course material, your post should be ***public***
  - But ***private*** if you need to post any of your source code or need to say something that reveals particular details or approaches of your solution

You can also discuss things during TA or instructor office hours

Policy on Late Work

Treat deadlines seriously

- Extensions will be given only under exceptional circumstances beyond your control
- Work in other courses or extracurricular activities are a normal part of life
- If you believe you will need an extension, ask ***in advance***, with detail, ***to Prof. Alan Cox (alc@rice.edu)***

Late homework assignments accepted up to two days late

- 10% penalty of original value of assignment for each day late
- A "day" is up to 24 hours or any fraction thereof

Late work beyond this is not accepted

Policy on Regrades

All regrade requests must be submitted within 7 days

- When you receive a graded homework assignment or exam, please carefully consider the feedback you received in the grading
- We encourage you to also discuss this with a TA or instructor in office hours
- If you do feel this homework assignment or some exam question was graded incorrectly, you must submit any regrade request ***within 7 days of when the grades were released***
- Regrade requests should be submitted ***by email to Prof. Alan Cox (alc@rice.edu)*** with ***subject "COMP 321 Regrade Request"***
- We will take whatever time is needed to ensure the grading is correct and fair
- That means your grade may go up, may go down, or may remain the same

Honor Code Policy

The Honor Code is a special privilege and responsibility at Rice University

> "As incoming students enter Rice, many are surprised by the degree to which the university's Honor Code extends trust to the student body. . . . The privileges of the Honor Code stem from the idea that Rice's aim is not just to instill knowledge in its students, but [to] also help them develop moral character. This idea is fundamental to Rice's identity: Students can and should be held to a high moral character standard."
>
> – Student editorial, ***The Rice Thresher***, January 20, 2016

Honor Code Policy

We take the Rice Honor Code very seriously and hope you do also

- Please carefully read the ***full*** Honor Code Policy in the syllabus, including
  - You may not use AI-based tools in this course, and you may not consult solutions from prior semesters of this or any similar course anywhere
  - You may not obtain code from any source other than the code we provide
- Note, you also may ***not*** place source code for any projects on any ***publicly*** accessible repository (such as a public GitHub repo)
  - Doing so gives aid to other students and thus would be an Honor Code violation, even if that is not your intent
  - Even after the semester, ***you may not make your code publicly available***

Unix Manual Pages

The Unix "man" pages document how to use and program Unix-like systems

- This is not a course about Unix (or Linux), but we will be programming on Linux in the labs and in the homework programming projects
- Each Unix-like system has documentation accessible online by the "man" command, divided into different sections

1	commands	man ls	man 1 printf
2	kernel calls	man execve	man 2 read
3	library functions	man 3 printf	man fprintf
etc.			


```
fork(2)                     System Calls Manual                     fork(2)

NAME
       fork - create a child process

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <unistd.h>

       pid_t fork(void);

DESCRIPTION
       fork()  creates  a new process by duplicating the calling process.  The
       new process is referred to as the child process.  The  calling  process
       is referred to as the parent process.

       The child process and the parent process run in separate memory spaces.
       At the time of fork() both memory spaces have the same content.  Memory
       writes,  file  mappings (mmap(2)), and unmappings (munmap(2)) performed
       by one of the processes do not affect the other.

       The child process is an exact duplicate of the  parent  process  except
       for the following points:

       *  The child has its own unique process ID, and this PID does not match
          the ID of any existing process group (setpgid(2)) or session.
```