Join on Zoom if you are remote (see website for link).

377 Operating Systems

01 Course Introduction


Welcome to
COMPSCI
377
Operating
Systems

Statement of Inclusivity

The staff for this course support the UMass commitment to diversity, and welcome individuals regardless of age, background, citizenship, disability, sex, education, ethnicity, family status, gender, gender identity, geographical origin, language, military experience, political views, race, religion, sexual orientation, socioeconomic status, and work experience. In this course, each voice in the classroom has something of value to contribute. Please take care to respect the different experiences, beliefs and values expressed by students and staff involved in this course.

Welcome!

Welcome to COMPSCI 377 Operating Systems!

Tim Richards (Instructor)

Tim Richards completed his Phd and Masters in Computer Science at UMass Amherst in 2010. His research work involved programming languages, compiler back-ends, operating systems, and verification of instruction set architectures. He has taught operating systems for a decade or so. He loves teaching all things related to systems!

Dave Dirnfeld (TA)

Bochen Xu (TA)

Calvin Chai (TA)

Sage Chircu (UCA)

Yichong Liu (UCA)

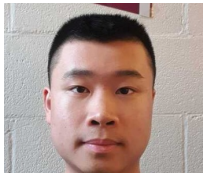
Ronan Salz (UCA)

Eugene Mak (UCA)

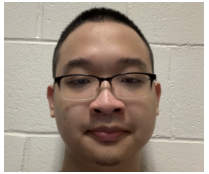
Meg Kaki (UCA)

Rohit Rangan (UCA)

Stuart Lustig (UCA)



Zhiyang Zuo (UCA)



Vinh Le (UCA)

Description

In this course we examine the important problems in operating system design and implementation. The operating system provides a well-known, convenient, and efficient interface between user programs and the bare hardware of the computer on which they run. The operating system is responsible for allowing resources (e.g., disks, networks, and processors) to be shared, providing common services needed by many different programs (e.g., file service, the ability to start or stop processes, and access to the printer), and protecting individual programs from one another. The course will start with a brief historical perspective of the evolution of operating systems over the last fifty years, and then cover the major components of most operating systems. This discussion will cover the tradeoffs that can be made between performance and functionality during the design and implementation of an operating system. Particular emphasis will be given to three major OS subsystems: process management (processes, threads, CPU scheduling, synchronization, and deadlock), memory management (segmentation, paging, swapping), file systems, and operating system support for distributed systems.

4 Credits

Prerequisites

- COMPSCI 230 Computer Systems Principles
- Or equivalent background

Major Applicability

- This course counts as a 300-level Computer Science Major elective.

Textbook

Operating Systems: Three Easy Pieces, Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dussea.

This is a free textbook available online. While the entire textbook is available for free from this site, if you so wish, a hard copy may be purchased on Amazon.

This book is absolutely fabulous as far as OS books go.

Software Platforms

We will be using a few software platforms as part of the management of this course.

- Moodle
- Piazza
- Zoom
- Echo360
- Gradescope
- Edlab
- VSCode

Moodle

Moodle is used to communicate course materials. You will be able to find the course syllabus, lecture material, recorded lectures, and other links on Moodle.

We also use Moodle for assignment scheduling, submission (links to assignments), and grades.

Piazza

- Piazza is used to communicate with the instructor and other students.
- You can ask questions, post answers, and discuss course material.
- You can also use Piazza to communicate with the instructor privately.

Zoom and Echo360

- All class meetings will be *synchronously* streamed on Zoom.
- All class meetings will be recorded by Zoom and uploaded to Echo360 at the end of class.
- You will be able to access the class on Zoom if you are unable to make it to class in-person.
- If you are unable to make it to class for a personal reason, you can view the recorded class meeting *asynchronously*.
- **DO NOT ABUSE THIS!**

Software Tools

We will use a variety of software tools in this course. The most important that you will need to install on your computer are:

- Visual Studio Code (VSCode)

You will be doing all of your coding in the Edlab environment. The Edlab environment is a cluster of Linux machines with all the installed software that you need.

You can use VSCode to remotely connect to the Edlab and code like you are on your own machine. Very nice.

Or, you can optionally use emacs, vim, or nano.

Lecture Format

01-LEC (44864): TuTh 10:00AM - 11:15AM, Goessmann Lab. Add rm 64

- Lectures will be led by the professor and provide a high-level overview of the course material.
- The presentation format will include a variety of slides, written notes, programming examples, activities, etc.
- You are expected to attend every lecture for your section and arrive promptly, so you do not disturb others.
- You may use electronic devices during class; however, its use must pertain to the activity at hand.

Discussion Format

01AA-DIS(44865): Fr 12:20PM - 1:10PM, Goessmann Lab. Add rm 64

- The discussion section is led by teaching assistants (TAs) for this course.
- There will also be undergraduate course assistants (UCA) assigned to your discussion section.
- You are expected to attend every discussion section.
- Discussions are used to begin discussion assignments and get help completing those assignments.

  Missing a discussion section does not excuse you from any activities that occur during that time. Do not ask to make up any missed work during discussion section time.

What happens if I can't attend a lecture or a discussion?

Although we want you to attend the lecture, we understand that sometimes that isn't always possible. For that reason, we provide flexible options that allow you to participate.

- **Remote Synchronous**: attend the lecture remotely on Zoom and ask questions through the chat system.
- **Asynchronous**: watch the recorded lecture at a time that is convenient for you.

Unless you have special arrangements, please do not assume that you can attend fully remote. We do want you in class to engage with the class!

Discussions are not zoomed/recorded. However, you must complete the discussion exercise.

Assignments and Grading

- (45%) Project Assignments
- (20%) Topic Quizzes
- (20%) Final Exam
- (15%) Discussion Exercises

(45%) Project Assignments

- There are approximately 6 projects that are hands on with programming as it relates to the material.
- You will be provided a “starter” project, but will require additions, modifications, or tasks to complete it.
- Most projects come with some tests (but not all) to guide you.
- You submit your projects to Gradescope which will run our auto-grader and apply additional private tests.

(20%) Topic Quizzes

- Topic quizzes will test your knowledge of the material.
- There will be approximately 6 of these assessments.
- They will be multiple choice, true/false, etc.
- They will be available on Moodle during a 24-hour period.
- Once you start a topic quiz you will have 50 minutes to complete it.

(20%) Final Exam

- There will be a final exam for this course that you will complete during the final examination period.
- The final exam will be cumulative and will include a certain percentage of the questions you have already seen on the topic quizzes.
- The final exam will also include some open response questions in addition to multiple choice, true/false, etc. style questions.
- You will have 2 hours to complete the final exam once you start.
- The exam will be “take home”

(15%) Discussion Exercises

- A discussion exercise is a short activity that will focus on expanding your knowledge in a particular topic area.
- They are often designed to provide guidance for an upcoming project or to enhance your understanding of the content for that week.
- They are completed on Gradescope.
- They are graded both automatically and manually.

Office Hours

- Office hours are times when we provide real-time access to the instructor, TAs, and UCAs.
- You do not need an appointment to attend office hours, attendance is optional, and all questions you have about the course are welcome.
- These sessions will be held at several different times during the week.
- Office hours will be posted on Moodle as soon as we are able to schedule them.
- Office hours will be both in-person and on Zoom

Accommodations

- The University of Massachusetts Amherst is committed to providing an equal educational opportunity for all students.
- If you have a documented physical, psychological, or learning disability on file with Disability Services (DS), you may be eligible for reasonable academic accommodations to help you succeed in this course.
- If you have a documented disability that requires an accommodation, please notify me as soon as possible so that we may make appropriate arrangements.
- For further information, please visit Disability Services (https://www.umass.edu/disability/).

Academic Honesty Pledge

Academic honesty is taken seriously as part of being a member of the CICS community. As an active member of this course you are required to sign a pledge that you will uphold the values and trust of our community and commit to honesty and integrity in all assignments, exams, and activities. This pledge also acknowledges the consequences of breaking this trust and accepting the results of your actions.

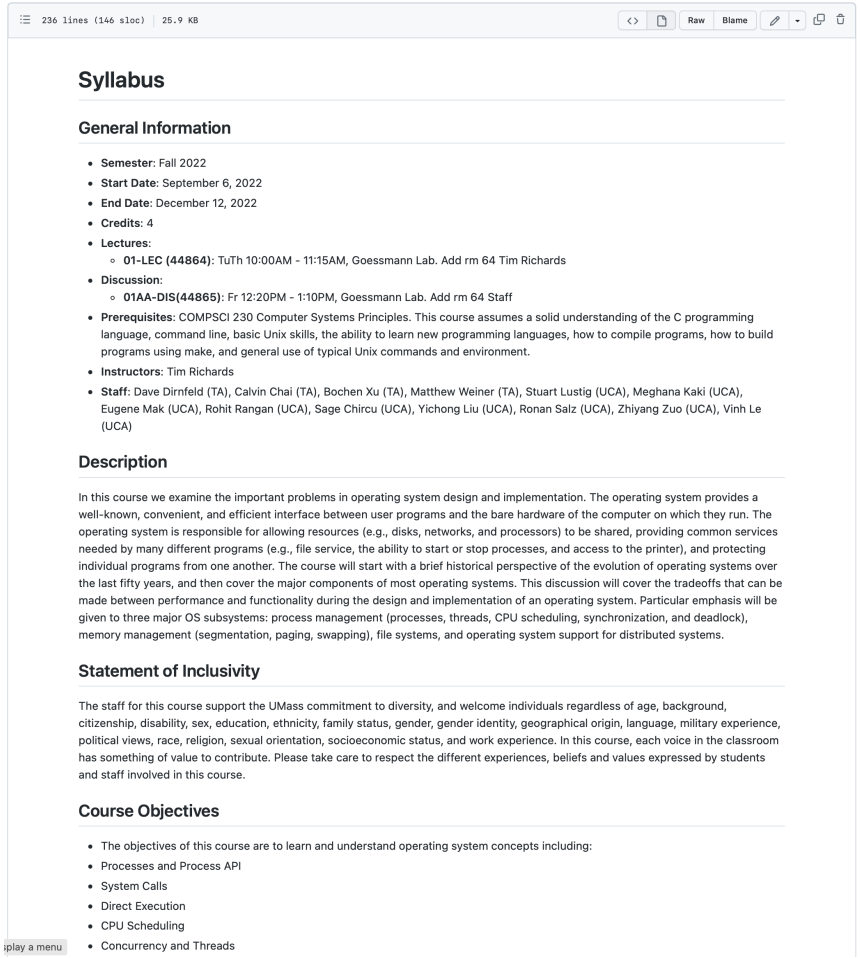
You must complete the pledge on Moodle. To stay in this course you must sign the pledge by answering YES. If you are unable to do so, you should withdraw from the course.

Syllabus

The **syllabus** is the primary resource for all details regarding the course

You should consult the syllabus for all aspects of the course

There may be circumstances where the syllabus needs to be updated, so always visit the syllabus online to ensure you are reading the most updated information



236 lines (146 sloc) | 25.9 KB

Raw Blame

Syllabus

General Information

- **Semester**: Fall 2022
- **Start Date**: September 6, 2022
- **End Date**: December 12, 2022
- **Credits**: 4
- **Lectures**:
  - **01-LEC (44864)**: TuTh 10:00AM - 11:15AM, Goessmann Lab. Add rm 64 Tim Richards
- **Discussion**:
  - **01AA-DIS(44865)**: Fr 12:20PM - 1:10PM, Goessmann Lab. Add rm 64 Staff
- **Prerequisites**: COMPSCI 230 Computer Systems Principles. This course assumes a solid understanding of the C programming language, command line, basic Unix skills, the ability to learn new programming languages, how to compile programs, how to build programs using make, and general use of typical Unix commands and environment.
- **Instructors**: Tim Richards
- **Staff**: Dave Dirnfeld (TA), Calvin Chai (TA), Bochen Xu (TA), Matthew Weiner (TA), Stuart Lustig (UCA), Meghana Kaki (UCA), Eugene Mak (UCA), Rohit Rangan (UCA), Sage Chircu (UCA), Yichong Liu (UCA), Ronan Salz (UCA), Zhiyang Zuo (UCA), Vinh Le (UCA)

Description

In this course we examine the important problems in operating system design and implementation. The operating system provides a well-known, convenient, and efficient interface between user programs and the bare hardware of the computer on which they run. The operating system is responsible for allowing resources (e.g., disks, networks, and processors) to be shared, providing common services needed by many different programs (e.g., file service, the ability to start or stop processes, and access to the printer), and protecting individual programs from one another. The course will start with a brief historical perspective of the evolution of operating systems over the last fifty years, and then cover the major components of most operating systems. This discussion will cover the tradeoffs that can be made between performance and functionality during the design and implementation of an operating system. Particular emphasis will be given to three major OS subsystems: process management (processes, threads, CPU scheduling, synchronization, and deadlock), memory management (segmentation, paging, swapping), file systems, and operating system support for distributed systems.

Statement of Inclusivity

The staff for this course support the UMass commitment to diversity, and welcome individuals regardless of age, background, citizenship, disability, sex, education, ethnicity, family status, gender, gender identity, geographical origin, language, military experience, political views, race, religion, sexual orientation, socioeconomic status, and work experience. In this course, each voice in the classroom has something of value to contribute. Please take care to respect the different experiences, beliefs and values expressed by students and staff involved in this course.

Course Objectives

- The objectives of this course are to learn and understand operating system concepts including:
- Processes and Process API
- System Calls
- Direct Execution
- CPU Scheduling
- Concurrency and Threads

iplay a menu

What’s An OS?

Interface between the user and the machine architecture.

Implements a virtual machine that is (hopefully) easier to program than raw hardware.

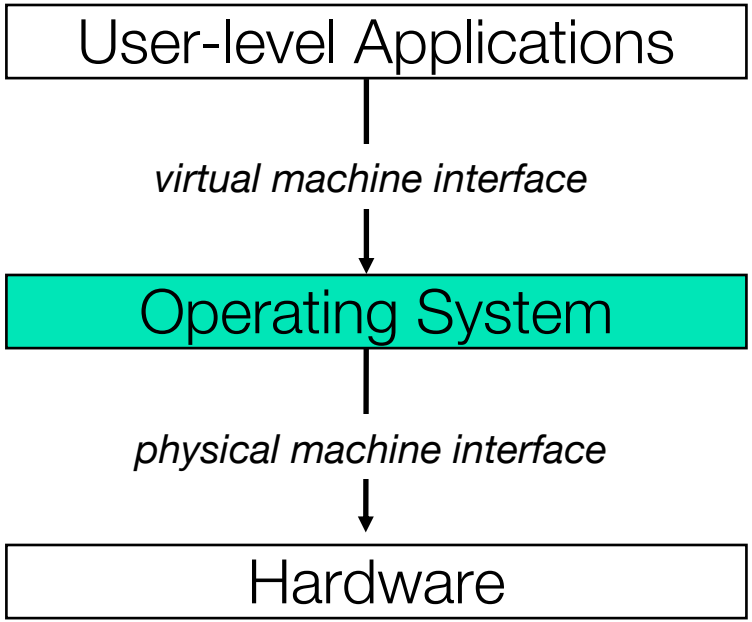
OS

Interface between user and architecture: Hides architectural details

Implements virtual machine: Easier to program than raw hardware

Illusionist Bigger, faster, reliable

Government: Divides resources, “Taxes” = overhead



Most Important Features

Services: The OS provides standard services (the interface) which the hardware implements.

- File system, virtual memory, networking, CPU scheduling, and time-sharing

Most Important Features

Services: The OS provides standard services (the interface) which the hardware implements.

- File system, virtual memory, networking, CPU scheduling, and time-sharing

Coordination: The OS coordinates multiple applications and users to achieve fairness and efficiency (throughput).

- Concurrency, memory protection, networking, and security.

Most Important Features

Services: The OS provides standard services (the interface) which the hardware implements.

- File system, virtual memory, networking, CPU scheduling, and time-sharing

Coordination: The OS coordinates multiple applications and users to achieve fairness and efficiency (throughput).

- Concurrency, memory protection, networking, and security.

Goal: Design an OS so that the machine is convenient to use (a software engineering problem) and efficient (a system and engineering problem).

Why Study OS?

Hilariously: You are unlikely to get a job building an OS.

Why Study OS?

Hilariously: You are unlikely to get a job building an OS.

However, *Understanding operating systems will enable you to use your computer more effectively*.

Why Study OS?

Hilariously: You are unlikely to get a job building an OS.

However, *Understanding operating systems will enable you to use your computer more effectively*.

Why does my application run slowly?

Why Study OS?

Hilariously: You are unlikely to get a job building an OS.

However, *Understanding operating systems will enable you to use your computer more effectively*.

Why does my application run slowly?

How can I make my application cheaper to run?

Why Study OS?

Hilariously: You are unlikely to get a job building an OS.

However, *Understanding operating systems will enable you to use your computer more effectively*.

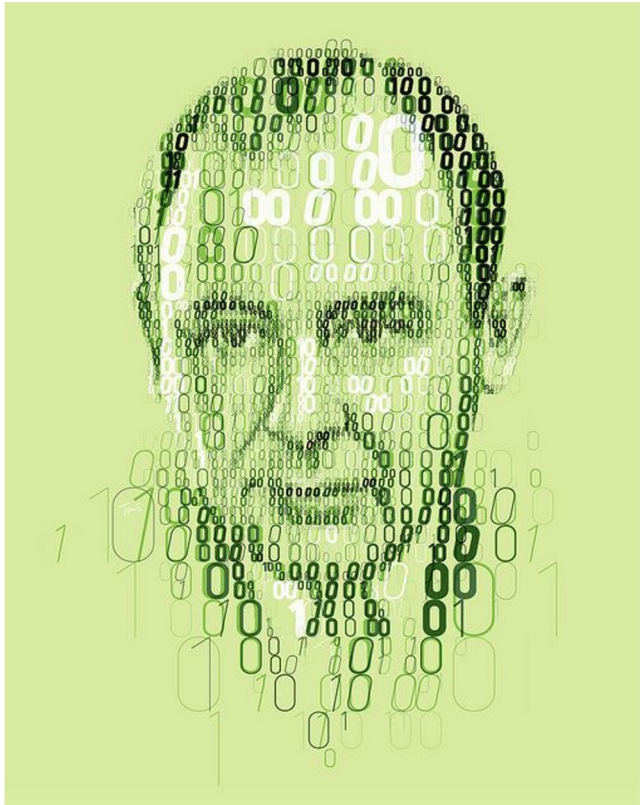
Why does my application run slowly?

How can I make my application cheaper to run?

It will also serve as an excellent example of system design issues whose results and ideas you will apply elsewhere.

Introduction to Operating Systems

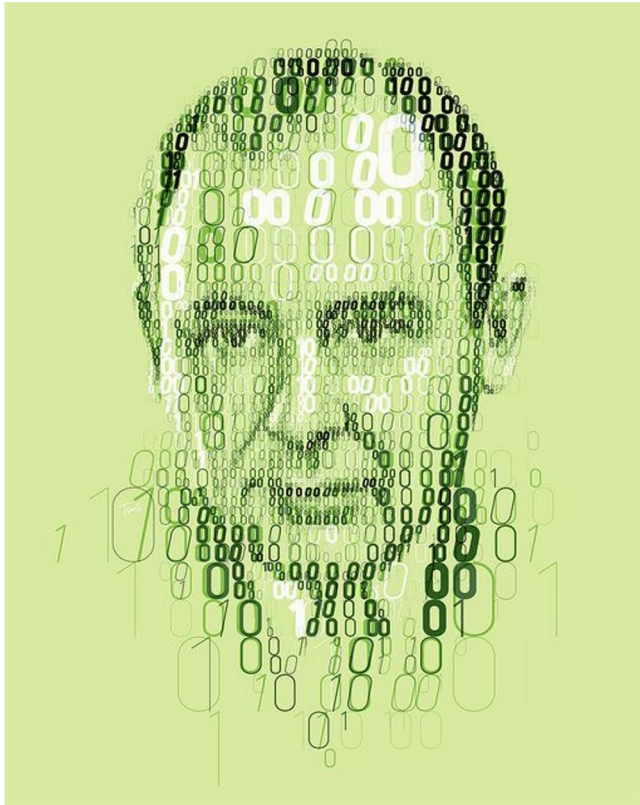
- **What happens when a program runs?**



Introduction to Operating Systems

- **What happens when a program runs?**
- **It does one simple thing:**

  Executes instructions!
  Many millions (or event billions) of times every second

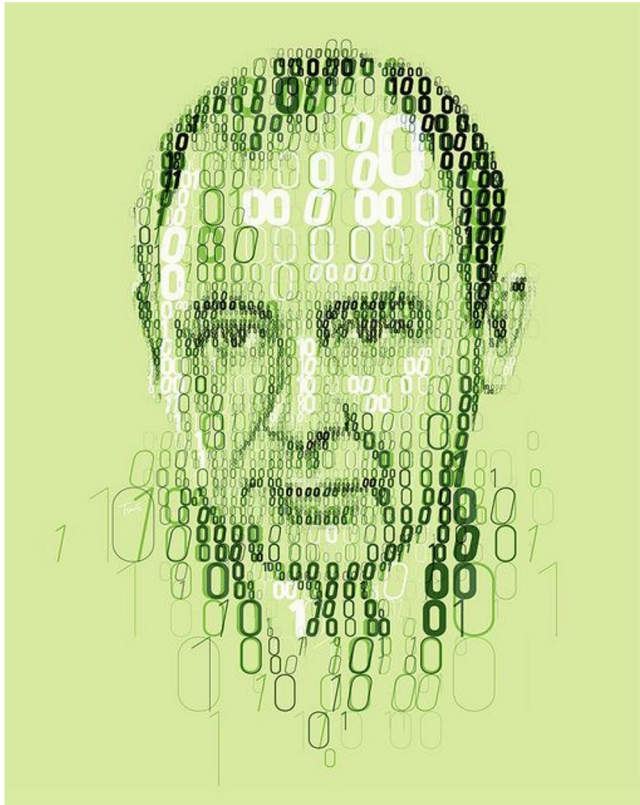


Introduction to Operating Systems

- **What happens when a program runs?**
- **It does one simple thing:**

  Executes instructions!
  Many millions (or event billions) of times every second

- **The processor**
  - Fetches an instruction
  - Decodes that instruction
  - Executes that instruction (add, subtract, move, load, store)
  - Checks conditions
  - Moves on to the next instruction and does it again



Introduction to Operating Systems

Thus, we have just described the Von Neumann model of computing.

Sounds simple, right?

But in this class, we will be learning that while a program runs…

lots of other *wild things* are going on with the primary goal of making the system easy to use.

Introduction to Operating Systems

There is a body of software that is responsible for making it easy to run programs.

Introduction to Operating Systems

There is a body of software that is responsible for making it easy to run programs.

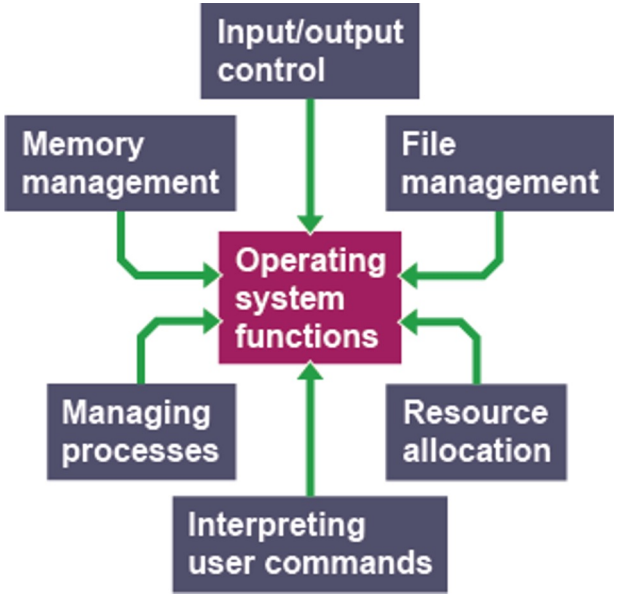
Allowing programs to share memory…

Enabling programs to interact with devices…

… and other fun stuff like that!

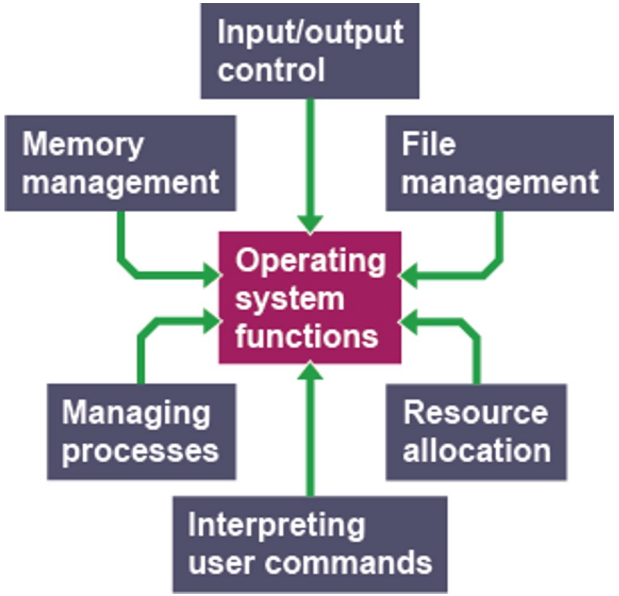
This body of software is called

the operating system (OS).



Virtualization

The primary mechanism the OS uses to accomplish this is virtualization.

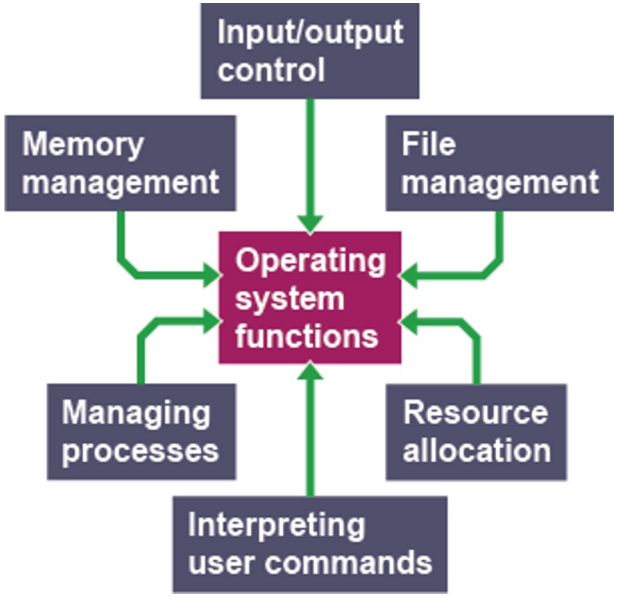


Virtualization

The primary mechanism the OS uses to accomplish this is virtualization.

The OS takes a physical resource (e.g., processor, memory, disk) and transforms it into a more general, powerful, and easy-to-use virtual form of itself.

Thus, we sometimes refer to the operating system as a *virtual machine*.



How to Virtualize Resources

THE CRUX OF THE PROBLEM: HOW TO VIRTUALIZE RESOURCES

One central question we will answer in this book is quite simple: how does the operating system virtualize resources? This is the crux of our problem. *Why* the OS does this is not the main question, as the answer should be obvious: it makes the system easier to use. Thus, we focus on the *how*: what mechanisms and policies are implemented by the OS to attain virtualization? How does the OS do so efficiently? What hardware support is needed?

How to Virtualize Resources

THE CRUX OF THE PROBLEM: HOW TO VIRTUALIZE RESOURCES

One central question we will answer in this book is quite simple: how does the operating system virtualize resources? This is the crux of our problem. *Why* the OS does this is not the main question, as the answer should be obvious: it makes the system easier to use. Thus, we focus on the *how*: what mechanisms and policies are implemented by the OS to attain virtualization? How does the OS do so efficiently? What hardware support is needed?

How to Virtualize Resources

THE CRUX OF THE PROBLEM: HOW TO VIRTUALIZE RESOURCES

One central question we will answer in this book is quite simple: how does the operating system virtualize resources? This is the crux of our problem. *Why* the OS does this is not the main question, as the answer should be obvious: it makes the system easier to use. Thus, we focus on the *how*: what mechanisms and policies are implemented by the OS to attain virtualization? How does the OS do so efficiently? What hardware support is needed?

How to Virtualize Resources

THE CRUX OF THE PROBLEM: HOW TO VIRTUALIZE RESOURCES

One central question we will answer in this book is quite simple: how does the operating system virtualize resources? This is the crux of our problem. *Why* the OS does this is not the main question, as the answer should be obvious: it makes the system easier to use. Thus, we focus on the *how*: what mechanisms and policies are implemented by the OS to attain virtualization? How does the OS do so efficiently? What hardware support is needed?

How to Virtualize Resources

THE CRUX OF THE PROBLEM: HOW TO VIRTUALIZE RESOURCES

One central question we will answer in this book is quite simple: how does the operating system virtualize resources? This is the crux of our problem. *Why* the OS does this is not the main question, as the answer should be obvious: it makes the system easier to use. Thus, we focus on the *how*: what mechanisms and policies are implemented by the OS to attain virtualization? How does the OS do so efficiently? What hardware support is needed?

How to Virtualize Resources

THE CRUX OF THE PROBLEM: HOW TO VIRTUALIZE RESOURCES

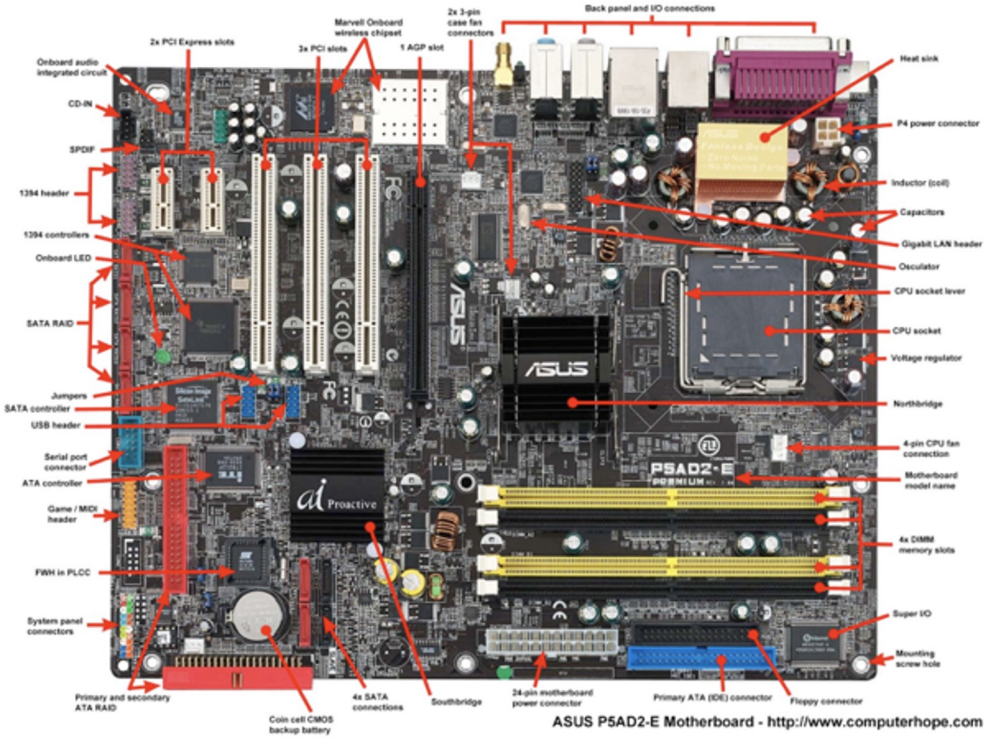
One central question we will answer in this book is quite simple: how does the operating system virtualize resources? This is the crux of our problem. *Why* the OS does this is not the main question, as the answer should be obvious: it makes the system easier to use. Thus, we focus on the *how*: what mechanisms and policies are implemented by the OS to attain virtualization? How does the OS do so efficiently? What hardware support is needed?

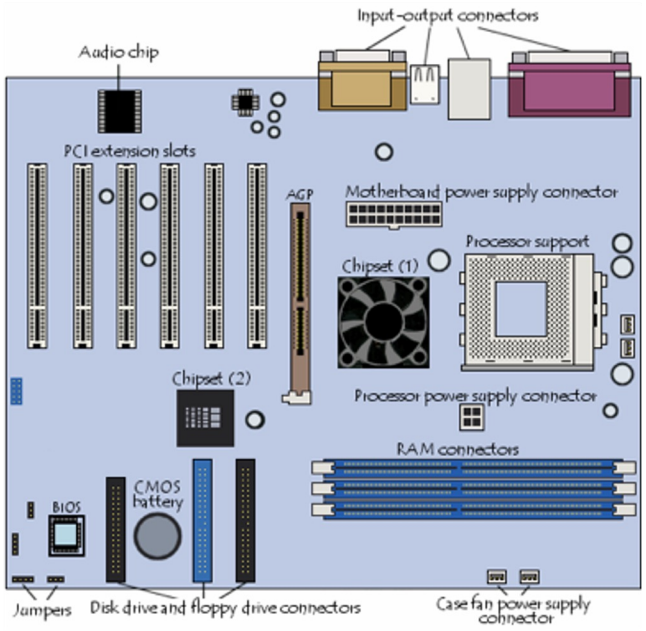
How to Virtualize Resources

THE CRUX OF THE PROBLEM: HOW TO VIRTUALIZE RESOURCES

One central question we will answer in this book is quite simple: how does the operating system virtualize resources? This is the crux of our problem. *Why* the OS does this is not the main question, as the answer should be obvious: it makes the system easier to use. Thus, we focus on the *how*: what mechanisms and policies are implemented by the OS to attain virtualization? How does the OS do so efficiently? What hardware support is needed?

What are we Virtualizing?






The End