CSE 120: Principles of Operating Systems
Lecture 1: Introduction

Prof. J. Pasquale
University of California, San Diego
January 7, 2019
Welcome

• In this class, we will explore
  – Basic concepts of operating systems (OS)
  – Design, implementation, structure
  – Principles that apply to all operating systems

• Today
  – Introductions
  – Class organization
  – Topic overview
Introductions: Teaching Staff

Instructor: Prof. J. Pasquale

Teaching Assistants
- Ujwal Bachiraju
- Walker Carlson
- Vijay Viswanath
- Zijing Xue

Tutors
- Ajeya Rengarajan
- Litao Qiao
- Tejas Gopal
- Ken Lin
- Zaki Siddiqui
- Lin Zhou
- Emily Chou
- Susmitha Kalidindi
- Shawn Gu
- Rohan Bhargava
- Curtis Tong
- Lihao He

© 2019 by Joseph Pasquale
Course Organization

- Lectures on Mondays and Wednesdays
- Discussion Sections on Fridays
- Programming Assignments: 4
- Exams: Midterm and Final
Lectures

• Lectures are key: Don’t miss them
  – Designed to motivate topics
  – Highlight what is most important

• Exam questions come directly from lectures
  – Lecture notes + what is said in class

• Book is important as a back-up reference
  – To fill in details
  – Provide an alternative explanation
Programming Assignments (PAs)

• Based on pedagogical OS: UMIX
  – UMIX = User-Mode UNIX
  – Developed specifically for this class

• 4 assignments on building parts of UMIX OS
  – Each builds on previous, of increasing complexity

• Only works on ieng9.ucsd.edu
  – If you are enrolled, you should have an account
  – Log in as soon as possible to make sure
UMIX Structure

User Processes

Running Program 1
Running Program 2
... Running Program n

UMIX

PA1: Context-Switching
PA2: Scheduling
PA3: Synchronization
PA4: Threads

UMIX CORE (libumix.a)

Hardware

© 2019 by Joseph Pasquale
UMIX Structure: Working on PA1

- **UMIX**: Context-Switching
- **PA1**: Scheduling
- **PA2**: Synchronizing
- **PA3**: Threads
- **UMIX CORE (libumix.a)**

User Processes:
- Running Program 1
- Running Program 2
- Running Program n
UMIX Structure: Working on PA2

User Processes

Running Program 1
Running Program 2
... Running Program n

UMIX

PA1: Context-Switching
PA2: Scheduling
PA3: Synchronization
PA4: Threads

UMIX CORE (libumix.a)

Hardware

© 2019 by Joseph Pasquale
All Work on ieng9 / cs120w

- To get credit, you must do **all** of your work:
  - on **ieng9** server (OK to remotely connect, e.g., ssh)
  - in your **cs120w** partition (prep cs120w)
  - in the **pa<N>** folder of your ieng9/cs120w account

```
[cs120w@ieng9]:~:$ getprogram1
  Creating Programming Assignment 1 directory
  Installing Programming Assignment 1's files
  Installation complete
[cs120w@ieng9]:~:$ ls
  CSE120-Results   pal
[cs120w@ieng9]:~:$ cd pal
[cs120w@ieng9]:pal:$ ls
  Makefile    mycode1.h    palb.c    pale.c    umix.h
  aux.h       pal.txt      palc.c    palf.c
  mycode1.c   pala.c       pald.c    sys.h
```
PA Deadlines

• Generally at 11:59PM on Sundays
  – PA1 Jan 13 end of week 1
  – PA2 Jan 27 end of week 3
  – PA3 Feb 17 end of week 6
  – PA4 Mar 10 end of week 9

• Note: ITS may not be available on weekend

• PA deadlines are absolute: *no exceptions*
  – Due to required UMIX updates for next PA
How Your PAs Are Graded

• Programs are graded by UMIIX Autograder
  – Runs a series of tests
  – At least one test per requirement in specification

• Results will be deposited in CSE120-Results
  – Output of test
  – Score

• You can appeal if you think score incorrect
Appealing Your PA Grade

• Must be based on your contentions that
  – Autograder made a mistake
  – Your program’s behavior is actually correct

• State clearly, explicitly, and in detail
  – What you believe the Autograder got wrong
  – Why your program’s behavior is actually correct

• You have 3 days to make an appeal
  – After that, no appeals will be accepted
After Your Appeal

• Your program will then be *manually* graded
  – You are liable for any/all problems found
  – Including ones missed by Autograder

• Clerical errors are still errors
  – A stray print statement can cause total failure
  – A bad character can cause a compile failure

• Can only grade what you hand in (no changes)

• Cost of appeal: 1 point (but can gain it back)
Collaboration Policy

- OK to discuss approaches, design, algorithms
- What is NOT OK: sharing any actual code
- What you submit must be your own work
- The only non-personally developed code you are allowed to use is what is provided to you via the getprogramN script
What Constitutes Cheating?

- Using ANY code that you did not come up with
  - other than what you’re given (official W19 copy)
- Producing *virtually* similar code
  - line-level similarity
  - even from common pseudo-code
  - factoring out renaming, moving, comments, ...
You Must Protect Your Code

• Making code available to others is cheating
  – Intentionally or unintentionally
  – Allowing your code to be viewed: not allowed

• It is your responsibility to protect your code

• You must not allow your account to be shared
No Code from External Sources

• Downloading code from websites is cheating
• Code \textit{derived} from such code is still cheating
• Any \textit{amount} of such code is cheating
  – even “just a few lines” is cheating
• Any \textit{kind} of such code is cheating
  – even “just a basic data structure” is cheating
• Any evidence of having downloaded code will be considered cheating, regardless of use
Consequences of Cheating

• You will be sent to Academic Integrity Office
• You will get an F if violation is confirmed
• Don’t expect a warning
• Don’t expect a second chance
• You may ask to withdraw your work, but only before it is graded; result is no credit (NQA)
• Checking will only be done at end of quarter
Exams

- Midterm will be in class, at mid-quarter (TBD)
- Final is on Saturday *before* start of finals week
- Both multiple-choice with justifications

17. As the quantum gets increasingly larger, round-robin scheduling performs more and more like (a) shortest remaining time first; (b) shortest job first; (c) first-come-first-served; (d) priority

**Question 17**  
A  B  C  D  E

**Justification:** the quantum increases, likelihood that a process will complete within a quantum increases. If each process completes before the next is run, this is effectively first-come-first-served.
Grading

• 4 Programming Assignments 30%
  – progressive in value: $3 + 6 + 9 + 12$
• Midterm Exam (insurance) 30% or 0%
• Final Exam 40% or 70%

• Your final grade will be based solely on above
• Grading is on an absolute scale
Resources

• Class website
  – cseweb.ucsd.edu/classes/wi19/cse120-a

• Piazza online discussion website
  – piazza.com/ucsd/winter2019/cse120/home

• Lecture notes
  – available on Piazza evening before lecture

• Computer system (for programming labs)
  – ieng9.ucsd.edu
Textbook

• Silberschatz, P. B. Galvin, G. Gagne: *Operating System Concepts*, 10\textsuperscript{th} Ed., Wiley, 2019

• If you are registered, also enrolled in eBook
  – Free first 2 weeks; then $48 \textit{automatically}
  – You can OPT OUT up to end of 2\textsuperscript{nd} week

• Lectures coordinated with this book

• Has good sample questions/answers
  – Some exam questions will come from these!
Advice For Getting a Good Grade

• For lecture material
  – Study book-related material *(after)* lecture
  – Make list of questions; use Piazza, office hours
  – Do lecture R&R (review and research) questions
  – Organize a study group

• Programming assignments
  – Start early (as soon as assignment comes out)
  – Do extensive testing – *testing is your responsibility*
Asking Questions is OK

• During lecture
  – I like questions, just raise your hand

• Piazza
  – All questions OK, except for solution code

• Office hours
  – Can go over any code and exam questions

• Email
  – Limited to private matters (and no code)
Piazza

• Online forum for asking / answering questions
• Depository for all class documents
  – Lectures
  – Programming assignment specifications
• To be enrolled on Piazza, you MUST:
  – register your FULL name: first AND last name
  – register your UCSD email account: xxx@ucsd.edu
  – be officially enrolled in the class
Etiquette on Piazza

• Before you post a question on Piazza
  – Is it answered in a lecture slide? ... In the book?
• When you ask a question
  – Try to be specific, focused, precise, clear
  – Attempt an answer, show your reasoning
  – Reference the relevant slide or pages in the book
• A good question will get a good fast response
• DO NOT POST SOLUTION CODE ON PIAZZA
We Want You to Get an A

• Everything is designed towards this goal
  – Lectures are continually revised, with added R&R
  – Programming assignments have hints, exercises
  – Exams based directly on lectures and PAs
  – Lots of help available via Piazza and office hours

• No curve: it’s possible everyone can get A

• Keys: take seriously, start early, get help
Any Questions?
Why Study Operating Systems?

• To know how your computer works
  – What may be wrong, how to enhance

• Systems programmers get respect
  – Understand OS → understand anything

• Operating systems is a classic area of CS
  – Roots in early 60’s, but still very relevant today

• Intellectually very challenging
  – Programming, structure, new technologies, ...
What is an Operating System?

• Software that makes computer easier to use
  — Broadly, for the user to interact with programs
  — For the programmer to use machine’s resources
  — Resources: CPU, memory, storage, I/O devices, ...

• Improves the computer’s capabilities
  — Performance: speed, efficiency
  — Reliability: correctness, fault tolerance
  — Security: privacy, authenticity, integrity
Operating System vs. the Kernel

• “Operating system” has many interpretations
  – E.g., all software on machine minus applications
• Our focus is much more limited: the *kernel*
  – All programs depend on it, accessed via sys calls
• Works closely with hardware
  – Accesses device registers, responds to interrupts
• Allocates basic resources
  – CPU time, memory space, use of I/O devices
Two Purposes of the Kernel

• To provide abstract machine
  – Interface for the programmer
  – Functions and resources
  – Goals: simplicity, convenience

• To manage resources
  – All the mechanisms and policies
  – Allocates usage: space and time
  – Goals: performance, reliability, security
Turn Undesirable into Desirable

• Undesirable inconveniences of reality ...
  – Complexity of hardware
  – Single/limited number of processors
  – Small/limited amount of memory

• Desirable conveniences: *illusions*
  – Simple, easy-to-use resources
  – Multiple/unlimited number of processors
  – Large/unlimited amount of memory
From Programmer’s Point of View

• Algorithm/program design is hard enough!
  – Allow programmer to focus on algorithm design
  – Not how to make machine run the algorithm

• Minimize accounting for computer limitations
  – Introduces unnecessary complexity
  – May lead to modifying the algorithm
  – May make the program not portable
Three Key Ideas

- **Abstraction**
  - *What* is the desired illusion

- **Mechanism**
  - *How* to create illusion: basic functionality
  - Fixed: works one way, the only way

- **Policy**
  - *Which* way to use mechanism, to meet a goal
  - Variable: many possible, select best for situation
Overview of Course Topics

• Resources: abstraction, mechanism, policy
  – CPU
  – Memory
  – Storage
  – I/O
• Operating system structure
• Advanced topics: security, distributed systems
Summary

• What is an operating system?
  – Software that is integral part of computer system
  – Makes it easy for user to use system
  – Keeps system running smoothly

• This course
  – Fundamental aspects of operating systems
  – Covering virtualization of CPU, memory, storage, I/O devices, and advanced topics
Textbook

• Chapters 1 and 2
  – Lecture-related: 1.1, 1.12, 2.1, 2.3, 2.8, 2.11
  – Hardware background: 1.2, 1.3
  – Recommended: 1.4-1.11, 2.2, 2.4-2.7, 2.9-2.10

• Do the sample exercises
  – Especially the ones with answers
  – I will likely use some for exams
Don’t Forget

• Activate your Piazza account for CSE 120
  – You must include your full name and UCSD email
• Check that your account on ieng9 is working
  – If not, contact ACMS – this is your responsibility!
• Get started on Programming Assignment 1
R&R: Review & Research

• These are questions that pertain to lecture
• Some review lecture concepts
• Some require research beyond the lecture
• Of varying levels of difficulty

  – Three stars ***  A  - most difficult
  – Two stars **  B
  – One star *  C
  – No star D  - least difficult
What is an operating system (OS)?
How does an OS make it easier to use?
What is an example of a difficulty a user would have in using a computer that had no OS?
What is the difference between a user and a programmer?
• What is meant by the term resource?*
• How does a computer’s OS improve its capabilities?**
• What is meant by a computer’s efficiency?**
• Why are speed and efficiency measures of performance?***
• How is speed different from efficiency?**
• What is meant by a computer’s correctness?*
R&R

• What does it mean for a computer system to be fault tolerant?*
• Why are correctness and fault tolerance aspects of reliability?**
• What is meant by each of the terms: privacy? authenticity? integrity?**
• Why are privacy, authenticity, and integrity considered aspects of security?**
R&R

• How is the *operating system* different from the *kernel*?

• What is an example of something that might be part of the operating system that would not be part of the kernel?*

• What are two actions a kernel can take that should not be allowed by a user program, and why should they not be allowed?**
R&R

• What does it mean to allocate a resource (given your earlier definition for resource)?*
• What is meant by *allocating CPU time*?**
• Why is meant by *allocating memory space*?**
• What about *memory time*: what is meant by this, and does it make sense to allocate it?***
• What about *CPU space*: what might be meant by this, and how might it be allocated?***
R&R

• What is meant by the term *abstract* (as in *abstract machine*)?**
• Why is the term *interface* relevant when discussing an abstract machine?**
• Why are functions and resources relevant when discussing an abstract machine?**
• What is a mechanism?*
• What is a policy?*
R&R

• Are mechanisms and policies aspects of an abstract machine: why or why not?***
• In what way are the goals, simplicity and convenience, as a class (i.e., not individually, but when viewed as a group and its characteristics), different from the goals, performance, reliability, and security, again when viewed as a class?***
R&R

• Why are the following considered to be “undesirable inconveniences of reality”:
  – complexity of hardware (provide an example)?**
  – limited number of processors? ...memory?*

• Why are the following considered “illusions”:
  – simple, easy-to-use resources?**
  – unlimited number of processors? ...memory?*
R&R

• Should concerns about the operating system enter into the programmer’s mind when designing and implementing a program: why or why not?**

• What are three examples of why the programmer should or shouldn’t take the operating system into account when designing a program?*
R&R

• What is meant by each of the following, and provide an example for each:
  – abstraction?*
  – mechanism?**
  – policy?**