

CS 111 – Operating Systems

- 1A Administrative Introduction
 - instructor, load and prerequisites
 - web site, syllabus, reading, and lectures
 - quizzes, exams, homework, projects
 - grading, academic honesty

1/1/2008

Introduction to Course and OS

1

Instructor: Mark Kampe (markk@cs.ucla.edu)

- Background
 - not regular faculty, not an academic
 - software engineer
 - 30 years of OS experience
 - developer, manager, architect & staff roles
 - here because I love OS and I love teaching
- Available for help
 - any time via e-mail
 - scheduled review sessions

1/1/2008

Introduction to Course and OS

2

Course Web Site

- <http://courseweb.seas.ucla.edu>
- schedules
 - reading, lectures, quizzes, exams, homework
- copies of lecture slides (pdf, 6 per page)
- announcements
- answers to homework, exam questions
- [Greek to English Dictionary](#)
- sample quiz, exam and final problems

1/1/2008

Introduction to Course and OS

3

Prerequisite Subject Knowledge

- CS 32 programming
 - objects, data structures, queues, stacks, tables, trees
- CS 33 systems programming
 - assembly language, registers, memory
 - linkage conventions, stack frames, register saving
- CS 118 networking
 - packets, addressing, routing, protocols,
 - protocol layering
- I will complement CS 151 coverage of
 - traps, interrupts, DMA

1/1/2008

Introduction to Course and OS

4

Course Format

- two weekly (average 40 page) reading assignments
 - mostly from the primary text
 - a few supplementary articles available on web
- two weekly (average 90 minute) lectures
 - each preceded by a quiz on the reading
 - broken into 20-45 minute segments
- one (60 minute) weekly chat session
 - where we can explore questions and implications
- weekly (1 hour) homework assignments
 - working with non-trivial concepts
- four (10-25 hour) team projects
 - exploring and exploiting OS features

1/1/2008

Introduction to Course and OS

5

Course Load

- reputations this course has
 - this is THE hardest class in the CS dept
 - fast pace through much non-trivial material
 - I try to provide a "rich" treatment of subject
- expectations you should have
 - lecture/lab 5-6 hours/week
 - reading/homework 4-8 hours/week
 - projects 3-20 hours/week
 - exam study 5-15 hours (twice)

1/1/2008

Introduction to Course and OS

6

Primary Text for Course

- Nutt: Operating Systems – third edition
 - background reading for virtually all lectures
- strengths for this course
 - good introduction to a good range of topics
 - good level of depth in most topics
 - good illustrations from real systems
- weaknesses for this course
 - occasional use of unnecessary formalisms
 - misses practical perspectives, implications

1/1/2008

Introduction to Course and OS

7

Course Lectures

- lectures will not
 - repeat material, well-covered in the reading
- lectures will
 - clarify and elaborate on the text
 - work through non-trivial examples
 - explore implications, applications
 - present material absent from the text
- all slides posted on-line
 - save you time taking notes
 - basis for exam study

1/1/2008

Introduction to Course and OS

8

I want you to ask questions

- lectures cover things I deem important
 - if I lecture on it, expect to see it on a test
- my goal is to help you master this material
 - if you had trouble, others probably did too
 - if a point is unclear, I didn't explain it properly
 - asking questions will help us all
- learn new concepts by interacting w/them
 - ask about implications and applications
 - challenge my generalities and simplifications

1/1/2008

Introduction to Course and OS

9

an Interactive On-line Course

- OS is a concept-rich course
 - non-interactive lectures are a problem
 - we must create interaction in other ways
- in-lecture questions
 - I pose questions during every lecture
 - typically about motivations and implications
 - pause the play-back, and consider the question
 - resume and I will discuss my answer
- on-line forum for questions and answers
- weekly chat sessions

1/1/2008

Introduction to Course and OS

10

Key Concepts

- this course is based on key concepts
 - they are the basis for the lectures
 - they are the basis for the exams
 - they are the basis for the projects
- the key concept list is on the web site
 - broken down, per lecture
 - use it in your exam review
 - be able to talk about each for five minutes

1/1/2008

Introduction to Course and OS

11

Course Grading

- breakdown of points
 - 18 daily quizzes 10%
 - 6-9 weekly homeworks 10%
 - 2 regular exams 30%
 - final exam 20%
 - projects 30%
- I do look at distribution for final grades
 - but expect the curve to fall near 90/80/70/60
- all scores available on MyUCLA
 - please check them for accuracy

1/1/2008

Introduction to Course and OS

12

Quizzes

- When: before each lecture
- Scope: reading assigned for that lecture
- Format:
 - 4 simple questions (definitions, examples, ...)
 - should require at most one sentence answer
- Goals:
 - test your familiarity with major concepts
 - force you to do reading prior to each lecture

1/1/2008

Introduction to Course and OS

13

Homework

- When: most weeks
- Scope: one week's reading & lectures
- Format:
 - 2-4 moderate essay questions or problems
 - some from the text, some I make up
- Goals:
 - test ability to apply key concepts from text
 - reinforce key concepts,
 - work where students often have problems

1/1/2008

Introduction to Course and OS

14

Regular Examinations

- When: 5th week, first half of final period
- Scope: 8-10 lectures of material
 - approximately 60% lecture, 40% text
- Format:
 - closed book
 - 10-15 essay questions, most with short answers
- Goals:
 - test understanding of key concepts
 - test ability to apply principles to practical problems

1/1/2008

Introduction to Course and OS

15

Final Exam

- When: second half of final exam period
- Scope: entire course
- Format:
 - 6-8 hard multi-part essay questions
 - you get to pick a subset of them to answer
- Goals:
 - test mastery of key concepts
 - test ability to apply key concepts to real problems
 - use key concepts to gain insight into new problems

1/1/2008

Introduction to Course and OS

16

Lab Projects

- Format:
 - 4-5 projects, of increasing difficulty
 - may be done solo or in teams
- Goals:
 - develop ability to exploit OS features
 - develop programming/problem solving ability
 - practice s/w project skills
- lab and lecture are fairly distinct
 - instructor cannot help you with projects
 - TA can't help w/lectures, homework, exams

1/1/2008

Introduction to Course and OS

17

Grading: partial and extra credit

- Partial Credit
 - will be awarded on all problems/projects:
 - clear understanding of problem
 - reasonable approach to problem
 - incomplete or flawed solutions
- Extra Credit
 - extra credit problems on exams
 - likely to be more difficult than the other problems
 - raise possible score above 100%
 - possible on any quiz/test/homework question
 - double points for any answer, better than mine

1/1/2008

Introduction to Course and OS

18

Late Assignments & Make-ups

- Quizzes
 - there are no make-ups
 - this would defeat their purpose
- Homework Assignments
 - not accepted after solutions are posted
- Exams
 - make-ups available after end of quarter
 - it will not be the same test

1/1/2008

Introduction to Course and OS

19

Academic Honesty

- it is OK to study with friends
 - discussing problems helps you to understand them
- it is OK to do independent research on a subject
 - there are many excellent treatments out there
- but all work you submit must be your own
 - do not write your homework with a friend
 - do not copy another student's work
 - do not turn in solutions from off the web
 - if you do research on a problem, cite your sources
- I decide when two assignments are too similar
 - and I forward them immediately to the Dean
- if you need help, ask the instructor

1/1/2008

Introduction to Course and OS

20

Academic Honesty – Projects

- do your own projects
 - work only with your team-mate
 - if you need additional help, ask the TA
- you must design and write all your own code
 - do not ask others how they solved the problem
 - do not copy solutions from the web, files or listings
 - cite any research sources you use
- protect yourself
 - do not show other people your solutions
 - be careful with old listings

1/1/2008

Introduction to Course and OS

21

CS 111 – Operating Systems

- 1B Introduction to the study of OS
 - relationship to other courses
 - why study operating systems
 - major themes & lessons in this course

1/1/2008

Introduction to Course and OS

22

Relationship to other courses

- OS will apply concepts from other courses
 - data structures, programming languages, assembly language programming, network protocols, computer architectures, ...
- OS prepares you for advanced courses
 - data bases and distributed computing
 - security, fault-tolerance, high availability
 - computer system modeling, queueing theory
- OS provides you with foundation concepts
 - processes, threads, virtual address space, files
 - capabilities, synchronization, leases, deadlock

1/1/2008

Introduction to Course and OS

23

Why Study Operating Systems?

- few of you will actually build OSs, but you will ...
 - set up, configure, manage computer systems
 - write programs that exploit OS features
 - work w/complex, distributed, parallel software
 - work with abstracted services and resources
- many hard problems have been solved in OS
 - synchronization, security, integrity, protocols, distributed computing, dynamic resource management, ...
 - in OS, we study these problems and their solutions
 - these approaches can be applied to other areas

1/1/2008

Introduction to Course and OS

24

Why I love Operating Systems

- they are extremely complex
 - yet must be understandable by one person
- they are very demanding
 - they require vision, imagination and insight
 - they must have elegance and generality
 - they demand meticulous attention to detail
- they are held to very high standards
 - performance, correctness, robustness,
 - scalability, extensibility, reusability
- they attract some very interesting people

1/1/2008

Introduction to Course and OS

25

Recurring Themes in OS

- view services as objects and operations
 - behind every object there is a data structure
- mechanism/policy separation
 - mechanism implements basic operations
 - policy controls allocation decisions
 - mechanisms shouldn't dictate/limit policies
 - change policies w/o changing mechanisms
- parallelism and asynchrony are powerful
 - but dangerous when used carelessly

1/1/2008

Introduction to Course and OS

26

Recurring Themes in OS

- an interface specification is a contract
 - responsibilities of producers & consumers
 - basis for product/release interoperability
- interface vs. implementation
 - an implementation is not a specification
 - many compliant implementations are possible
 - inappropriate dependencies cause problems
- modularity and functional encapsulation
 - complexity hiding and appropriate abstraction

1/1/2008

Introduction to Course and OS

27

Life Lessons in this Class

- There Aint No Such Thing As A Free Lunch!
 - everything has a cost, there are always trade-offs
- Keep It Simple, Stupid!
 - avoid complex solutions, and being overly clever
 - they usually create more problems than they solve
- Be very clear what your goals are
 - make the right trade-offs, focus on the right problems
- Responsible and Sustainable living
 - understand the consequences of our actions
 - nothing must be lost, everything must be recycled
 - it is all in the details

1/1/2008

Introduction to Course and OS

28

CS 111 – Operating Systems

- 1C - What is an Operating System
 - what OS does, how it appears to its clients
 - abstracted resources
 - simplifying, generalizing
 - serially reusable, partitioned, sharable

1/1/2008

Introduction to Course and OS

29

What Does an OS do?

- manage hardware for programs
 - allocate hardware and manage its use
 - enforce controlled sharing (and privacy)
 - oversee execution and handle problems
- abstract the hardware
 - make it easier to use, improve s/w portability
 - optimize performance
- new abstractions to enable applications
 - powerful features beyond the bare hardware

1/1/2008

Introduction to Course and OS

30

How OS appears to clients

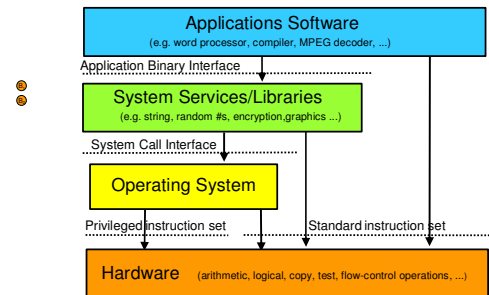
- management & abstraction services
 - invisible, they happen behind the scenes
- applications see objects and their services
 - CPU supports data-types and operations
 - bytes, shorts, longs, floats, pointers, ...
 - add, subtract, copy, compare, indirection, ...
 - so does an Operating System
 - files, processes, threads, devices, ports, ...
 - create, destroy, read, write, signal, ...
- OS extends a computer
 - creating a much richer virtual computing platform
 - supporting richer objects, more powerful operations

1/1/2008

Introduction to Course and OS

31

HW/SW Platform Layering



1/1/2008

Introduction to Course and OS

32

What makes the OS so special?

- it is always in control of the hardware
 - automatically loaded when the machine boots
 - first software to have access to hardware
 - continues running while apps come & go
- it alone has complete access to hardware
 - privileged instruction set, all of memory & I/O
- mediates applications' access to hardware
 - block, permit, or modify application requests
- it is trusted
 - to store and manage critical data
 - to always act in good faith

1/1/2008

Introduction to Course and OS

33

What functionality is in OS?

- as much as necessary, as little as possible
 - OS code is very expensive to develop and maintain
- functionality must be in the OS if it ...
 - requires the use of privileged instructions
 - requires the manipulation of OS data structures
 - must maintain security, trust, or resource integrity
- functions should be in libraries if they ...
 - are a service commonly needed by applications
 - do not actually have to be implemented inside OS
- but there is also the performance excuse
 - some things may be faster if done in the OS

1/1/2008

Introduction to Course and OS

34

Abstracted Resources

- Map one set of objects/operations into another
 - e.g. CPU->processes, disk->files, network->services
- simplifying abstractions
 - easier to use than the original resources
 - compartmentalize/encapsulate complexity
 - eliminate behavior that is irrelevant to user
 - create more convenient behavior
- examples:
 - a GPS moving-map navigation system
 - a network protocol that provides a reliable channel

1/1/2008

Introduction to Course and OS

35

Abstracted Resources

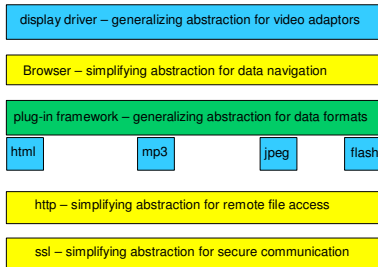
- generalizing abstractions
 - make many different types appear to be same
 - enable applications to deal with single common class
 - usually involves a common unifying model
 - portable document format (pdf) for printers
 - SCSI standard for disks, CDs and tapes
 - usually involves a federation framework
 - per sub-type implementations of standard functions
- examples
 - printer drivers make different printers look the same
 - browser plug-ins to handle multi-media data

1/1/2008

Introduction to Course and OS

36

Layers of abstraction – browser



1/1/2008

Introduction to Course and OS

37

General Types of Resources

- Serially reusable resources (time multiplexing)
 - used by multiple clients, but only one at a time
 - require access control to ensure exclusive use
 - require graceful transitions from one user to the next ●
 - examples: printers, bathroom stalls

1/1/2008

Introduction to Course and OS

38

General Types of Resources

- Partition-able resources (spatial multiplexing)
 - divided into pieces for multiple clients
 - needs access control to ensure containment/privacy ●
 - examples: disk space, dormitory rooms

1/1/2008

Introduction to Course and OS

39

General Types of Resources

- Sharable Resources
 - usable by multiple concurrent clients
 - clients do not have to “wait” for access to resource
 - clients do not “own” a particular subset of resource
 - may involve (effectively) limitless resources
 - air in a room, shared by occupants
 - copy of the operating system, shared by processes
 - may involve under-the-covers multiplexing ●
 - cell-phone channel (time and frequency multiplexed)
 - shared network interface (time multiplexed)

1/1/2008

Introduction to Course and OS

40

CS 111 – Operating Systems

- 1D – The Evolution of Operating Systems
 - early computers, batch processing
 - time sharing
 - embedded systems, work stations, PCs
 - client/server computing

1/1/2008

Introduction to Course and OS

41

OS Evolution – early computers

- usage
 - scheduled for use by one user at a time
- input
 - paper cards, paper tape, magnetic tape
- output
 - paper cards, paper tape, print-outs, magnetic tape
- software
 - compilers, assemblers, math packages
 - no “resident” operating system
- debugging
 - in binary, via lights and switches

1/1/2008

Introduction to Course and OS

42

Evolution – Batch Processing

- typified by the IBM System/360 (mid 1960s)
 - programs submitted and picked up later
 - input and output spooling to tape and disk
- goals: efficient CPU use, maximize throughput
 - I/O able to proceed with minimal CPU
 - overlapped execution and I/O maximize CPU usage
 - limited multi-tasking ability to minimize idle time
- software
 - batch monitor ... to move from one job to the next
 - I/O supervisor ... to manage background I/O
- debugging (in hex or octal via paper core dumps)

1/1/2008

Introduction to Course and OS

43

Evolution – Time Sharing

- Typified by IBM/CMS, Multics, UNIX
 - multi-user, interaction through terminals
 - all programs and data stored on disk
- goals: sharing for interactive users
 - interactive apps demand short response time
 - enhanced security required to ensure privacy
- OS and system services expanded greatly
 - terminal I/O, synchronization, inter-process communication, networking, protection, etc.

1/1/2008

Introduction to Course and OS

44

Evolution – Embedded Systems

- general purpose systems vs. appliances
 - running software vs. performing a service
- many appliances based on computers
 - video games, CD players, TV signal decoders
 - telephone switches, avionics, medical imaging
- appliances require increasingly powerful OSs
 - multi-tasking, networking, plug-n-play devices
- G.P. OS becoming more appliance-like
 - ultra-high availability, more automation,
 - easier to use, less management intensive

1/1/2008

Introduction to Course and OS

45

Evolution: PCs & Work Stations

- PCs returned to single user paradigm
 - initially minimal I/O and system services
 - file systems & interactivity from timesharing systems
- advent of personal productivity applications
 - high end applications gave rise to work-stations
- advent of local area networking
 - file transfer and e-mail led to group collaboration
 - the evolution of work groups and work-group servers
- PCs got bigger, faster, more services
 - they became indistinguishable from work stations

1/1/2008

Introduction to Course and OS

46

Evolution: network is the computer

- Client/Server Computing
 - centralized file and print servers for work groups
 - centralized mail, database servers for organizations
 - clients got thinner, servers became necessary
- Wide-Area Networking
 - e-mail, HTML/HTTP and the world-wide-web
 - electronic business services
- Distributed Computing Platforms
 - services are offered by/among groups of systems
 - system services must enable distributed applications

1/1/2008

Introduction to Course and OS

47

Evolution of Operating Systems

- they have grown larger and more sophisticated
- their role has fundamentally changed
 - from shepherding the use of the hardware
 - to shielding the applications from the hardware
 - to providing powerful application computing platform
- they still sit between applications and hardware
- best understood through services they provide
 - capabilities they add
 - applications they enable
 - problems they eliminate

1/1/2008

Introduction to Course and OS

48

For Next Lecture

- read chapters 1 through 3
 - a lot of pages, but the material is very simple
 - WARNING:
 - obtuse fork/join example in 2.3
 - see description in [Greek to English Dictionary](#)
- there will be a quiz on this material