DAT320
Operating Systems

Fall 2013
Course Introduction

University of
Stavanger

Today

- Class overview
- What is an operating system?

Personnel

- Instructor
  - Hein Meling, Office: E424
  - Office hours: Just come by the office
- Teaching Assistants
  - Morten Mossige, Office: E440
  - Office hours: (Monday)/Tuesday

Pre-requisites

- Advanced Programming skills
  - Java, C#, C/C++
- Computer Networking
  - Programming with TCP and UDP

Course Web Site

- https://github.com/uis-dat320-fall2014/
  - (Syllabus)
  - Lecture plan
  - Reading material
  - Lab projects
  - (but no lecture notes)

Syllabus

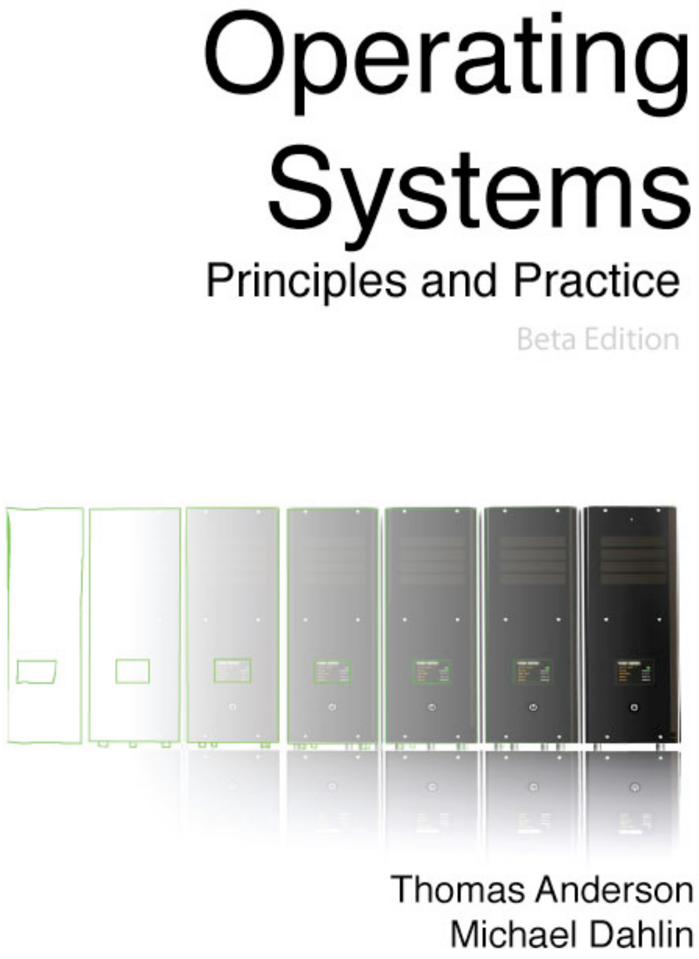
- Chapters from textbook
- Home work exercises from text book
- Additional papers (to be defined)
- Lab project
- Handouts/slides (published throughout the semester)

Home Work Exercises

- I will give some HW exercises from the book
- Reinforce lecture material…no better practice for exam
  - No handins, no grading
  - No solution (yet, but maybe in the future)
  - Can also practice on previous exams

Textbook

- Recursive Books, Ltd., Beta Edition
- News editions may work too!



Operating Systems

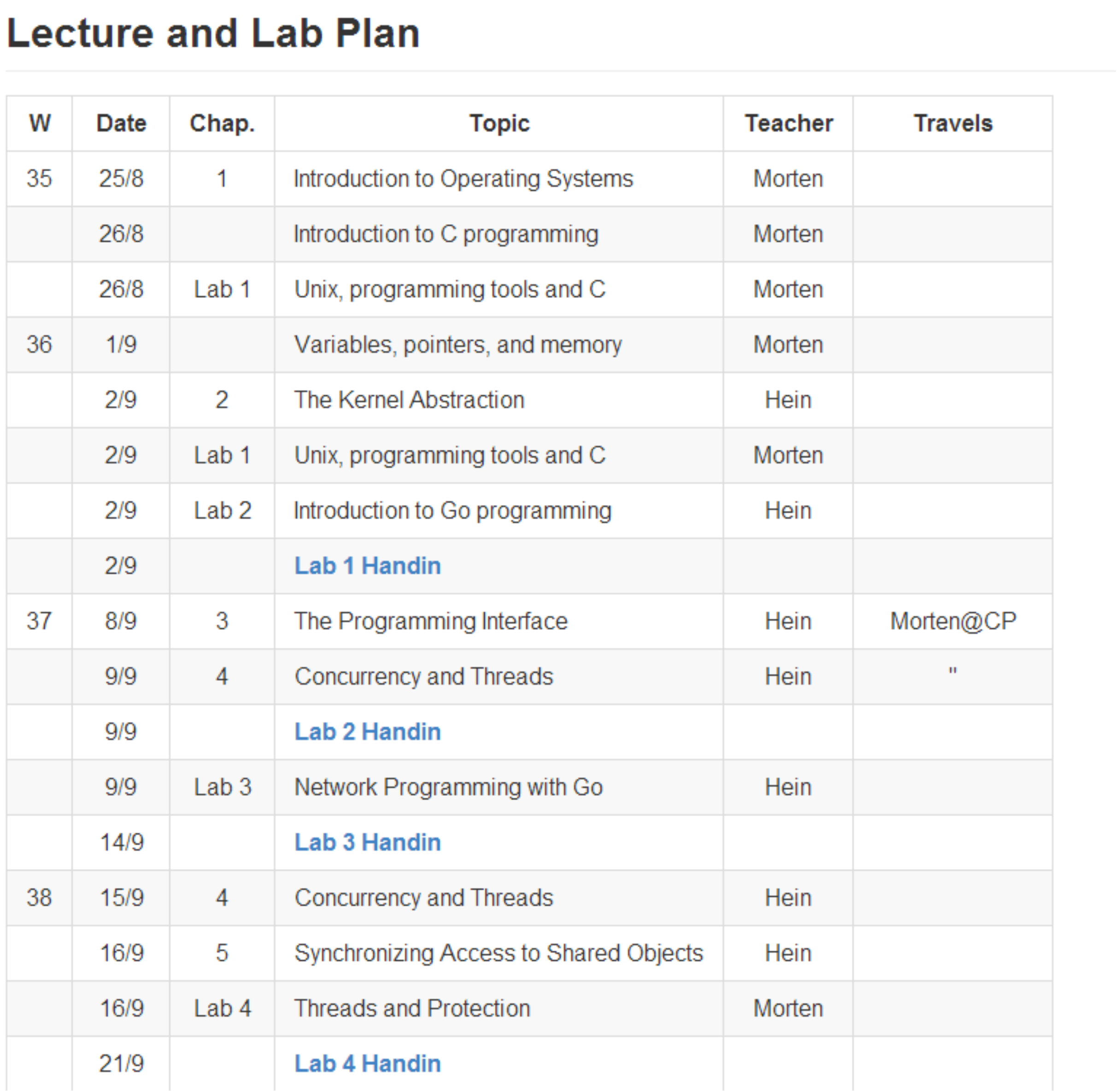
Principles and Practice

Beta Edition

Thomas Anderson
Michael Dahlin

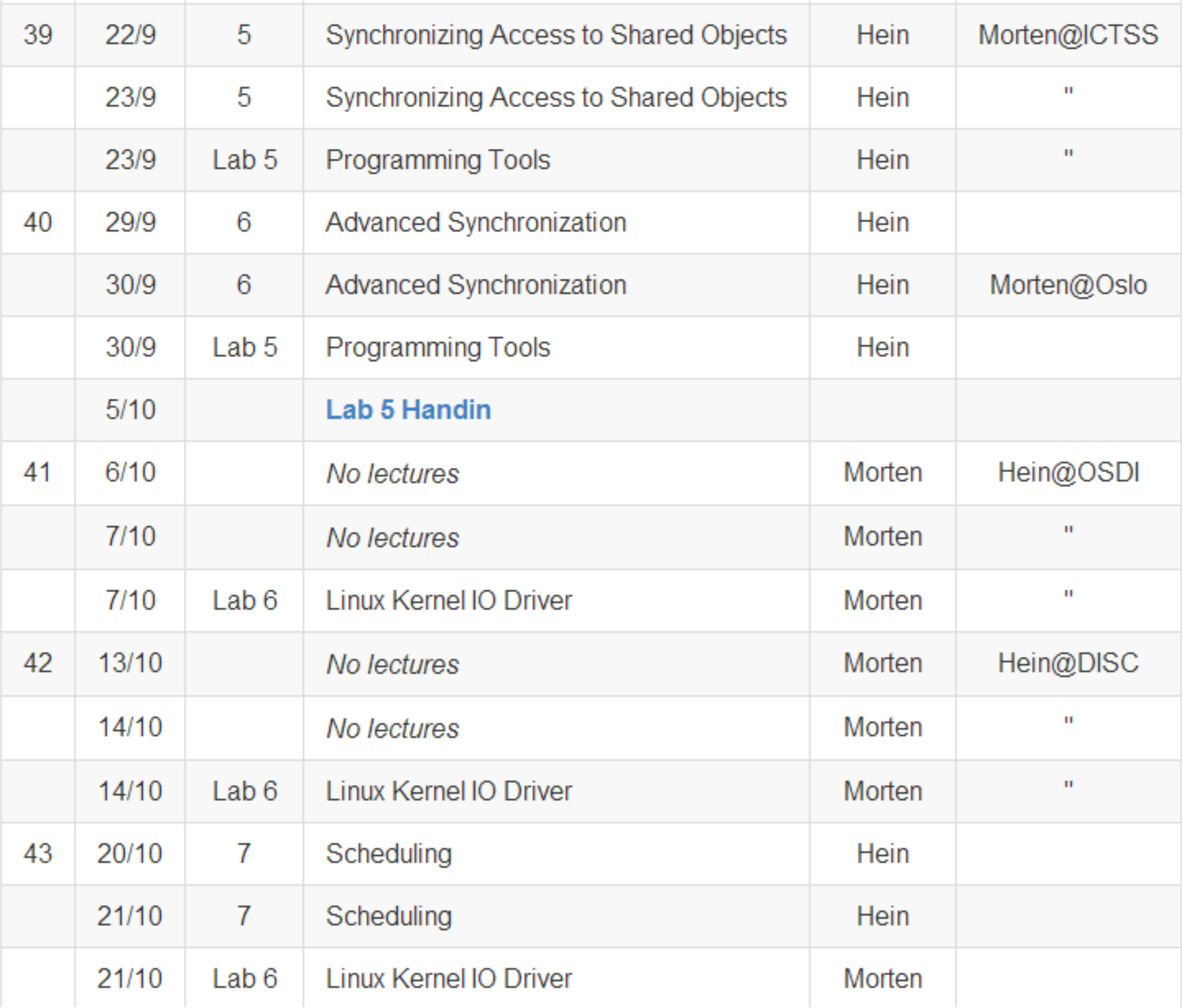
Lectures

- Mostly black board
- Only rarely slides
- Monday 0815-1100 D205
- Tuesday 0815-1100 D205

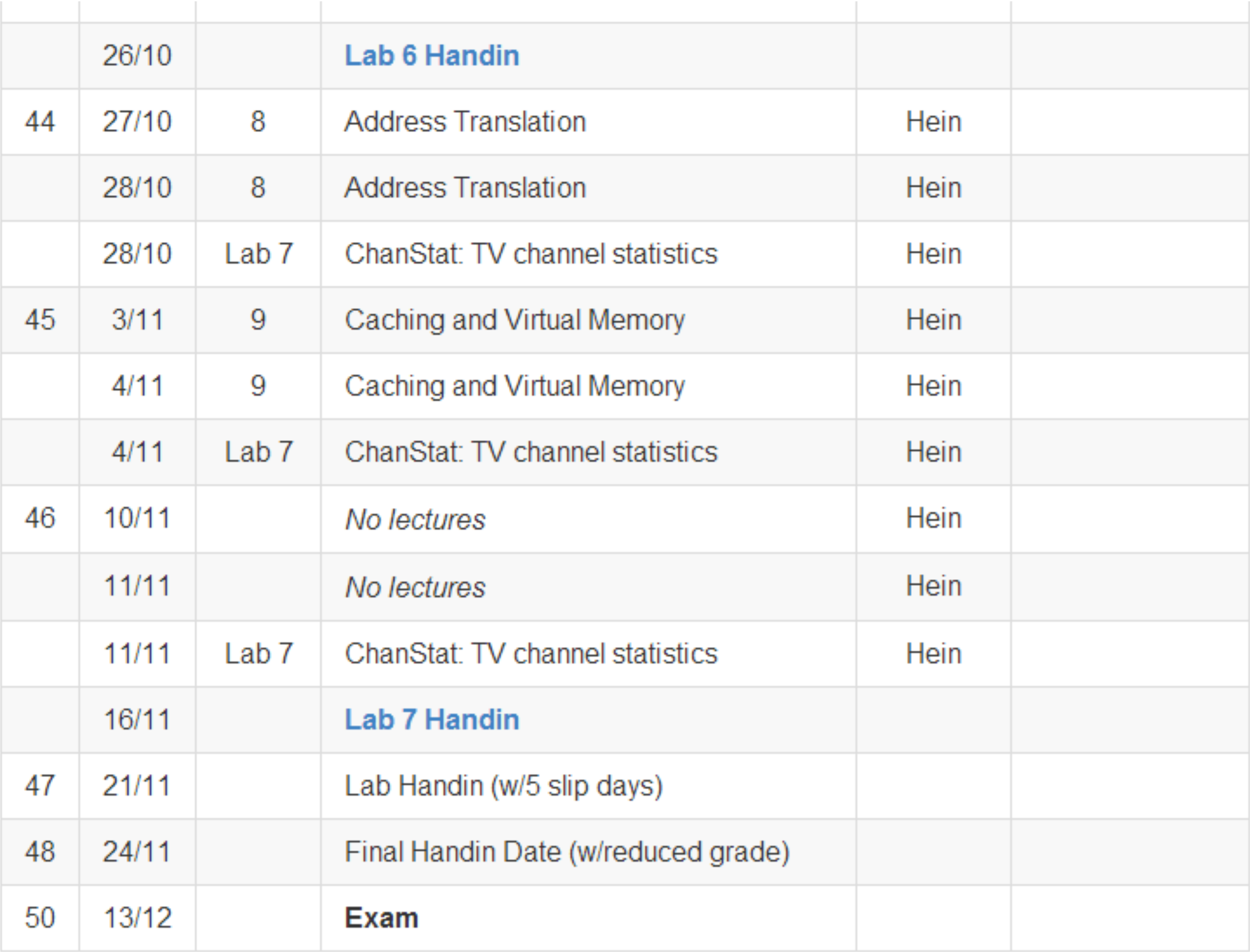


Lecture and Lab Plan

W	Date	Chap.	Topic	Teacher	Travels
35	25/8	1	Introduction to Operating Systems	Morten	
	26/8		Introduction to C programming	Morten	
	26/8	Lab 1	Unix, programming tools and C	Morten	
36	1/9		Variables, pointers, and memory	Morten	
	2/9	2	The Kernel Abstraction	Hein	
	2/9	Lab 1	Unix, programming tools and C	Morten	
	2/9	Lab 2	Introduction to Go programming	Hein	
	2/9		**Lab 1 Handin**		
37	8/9	3	The Programming Interface	Hein	Morten@CP
	9/9	4	Concurrency and Threads	Hein	"
	9/9		**Lab 2 Handin**		
	9/9	Lab 3	Network Programming with Go	Hein	
	14/9		**Lab 3 Handin**		
38	15/9	4	Concurrency and Threads	Hein	
	16/9	5	Synchronizing Access to Shared Objects	Hein	
	16/9	Lab 4	Threads and Protection	Morten	
	21/9		**Lab 4 Handin**		



39	22/9	5	Synchronizing Access to Shared Objects	Hein	Morten@ICTSS
	23/9	5	Synchronizing Access to Shared Objects	Hein	"
	23/9	Lab 5	Programming Tools	Hein	"
40	29/9	6	Advanced Synchronization	Hein	
	30/9	6	Advanced Synchronization	Hein	Morten@Oslo
	30/9	Lab 5	Programming Tools	Hein	
	5/10		**Lab 5 Handin**		
41	6/10		*No lectures*	Morten	Hein@OSDI
	7/10		*No lectures*	Morten	"
	7/10	Lab 6	Linux Kernel IO Driver	Morten	"
42	13/10		*No lectures*	Morten	Hein@DISC
	14/10		*No lectures*	Morten	"
	14/10	Lab 6	Linux Kernel IO Driver	Morten	"
43	20/10	7	Scheduling	Hein	
	21/10	7	Scheduling	Hein	
	21/10	Lab 6	Linux Kernel IO Driver	Morten	



	26/10		**Lab 6 Handin**		
44	27/10	8	Address Translation	Hein	
	28/10	8	Address Translation	Hein	
	28/10	Lab 7	ChanStat: TV channel statistics	Hein	
45	3/11	9	Caching and Virtual Memory	Hein	
	4/11	9	Caching and Virtual Memory	Hein	
	4/11	Lab 7	ChanStat: TV channel statistics	Hein	
46	10/11		*No lectures*	Hein	
	11/11		*No lectures*	Hein	
	11/11	Lab 7	ChanStat: TV channel statistics	Hein	
	16/11		**Lab 7 Handin**		
47	21/11		Lab Handin (w/5 slip days)		
48	24/11		Final Handin Date (w/reduced grade)		
50	13/12		**Exam**		

D

Lab Projects

Lab

- We will use the Linux lab in E353
  - Roughly 20 machines
  - You can also use your home machine/ laptop
  - Barebones VirtualBox image with Linux and Go setup
- There will be 7 mandatory lab handins
- 5 slipdays!

Slip days

- Totally five slipdays
- For each extra slip day used, your grade is reduced by 5 points (0-100).
- **Resubmission (but only one pr. handin):**
  - 5 points for the first resubmission
  - 10 points for the second resubmission
  - 20 points for the third resubmission
  - A fourth resubmission will not be allowed, resulting in failing the lab.

Lab Hours

- Lab E353 is reserved Tuesdays
  - 0815-1600
  - 1115-1400 with TA, ex lunch
  - May use lab at other times, except when reserved for other uses

Unix Account Registration

- To complete the lab you'll need:
  - A Unix user account
  - Goto: http://user.ux.uis.no/ today!
  - Register today before 15:00 to get access for tomorrows lab.
  - To get physical access to the lab you'll need:
    - User account
    - Pincode: 2244

Group Registration

- The main lab parts will be carried out in groups
  - General rule: group of two students
  - Working alone or three member groups only accepted by application stating a reason
- Send group registration info by email to Morten
  - Names and email addresses
  - Tag email subject with [DAT320] Group Registration

Lab Project Parts

1. Unix tools, VI and C
2. Go intro
3. Network Programming in Go
4. Threads and Protection
5. Go Programming tools
6. Kernel Driver for IO
7. ChanStat: TV Channel Statistics

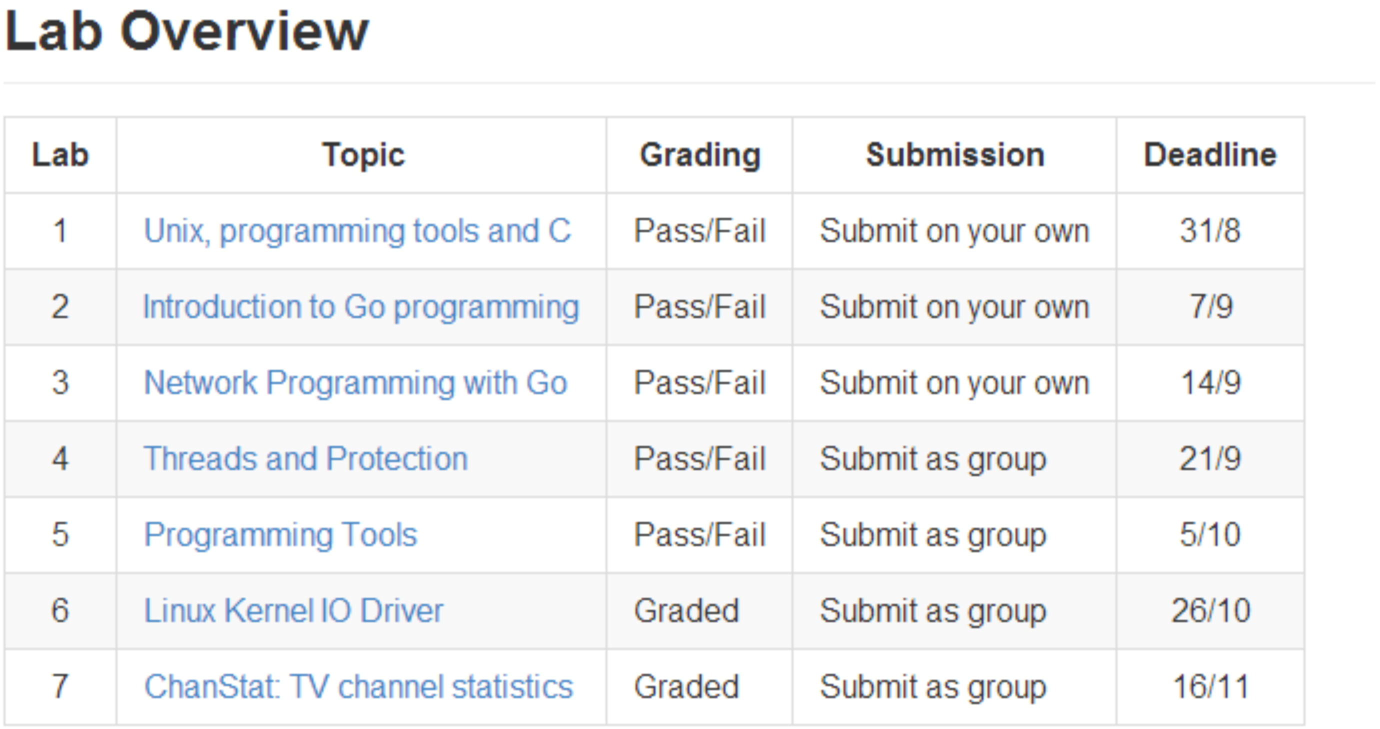
Lab Project Parts

Pass/Fail

1. Unix tools, VI and C
2. Go intro
3. Network Programming in Go
4. Threads and Protection
5. Go Programming tools

Graded

6. Kernel Driver for IO
7. ChanStat: TV Channel Statistics



Lab Overview

Lab	Topic	Grading	Submission	Deadline
1	Unix, programming tools and C	Pass/Fail	Submit on your own	31/8
2	Introduction to Go programming	Pass/Fail	Submit on your own	7/9
3	Network Programming with Go	Pass/Fail	Submit on your own	14/9
4	Threads and Protection	Pass/Fail	Submit as group	21/9
5	Programming Tools	Pass/Fail	Submit as group	5/10
6	Linux Kernel IO Driver	Graded	Submit as group	26/10
7	ChanStat: TV channel statistics	Graded	Submit as group	16/11

Why Go?

- Go is low-level, but garbage collected
- Concurrency (goroutines and channels)
- Fast and brief
- We are developing software for fault tolerant pub/sub in Go
- Start today:
    - http://golang.org/

Linux Kernel Driver

- Morten will give a brief intro to C/C++ programming in lab hour at a later time
  - Not a full C language course; some self-study is necessary
- Also explain the basics of kernel driver programming

Grading Policy

- Final exam: 60%
- Lab projects: 40%
  - Oral examination in the lab
  - Grade based on handins and examination
  - Must pass all labs to attend exam
  - More details will follow later
  - https://github.com/uis-dat320-fall2014/course-info/blob/master/labpolicy.md

Academic World of Operating Systems Research

- Symposium on Operating Systems Principles (SOSP)
- Symposium on Operating Systems Design and Implementation (OSDI)
- ACM Transactions on Computer Systems

Learning a new programming language

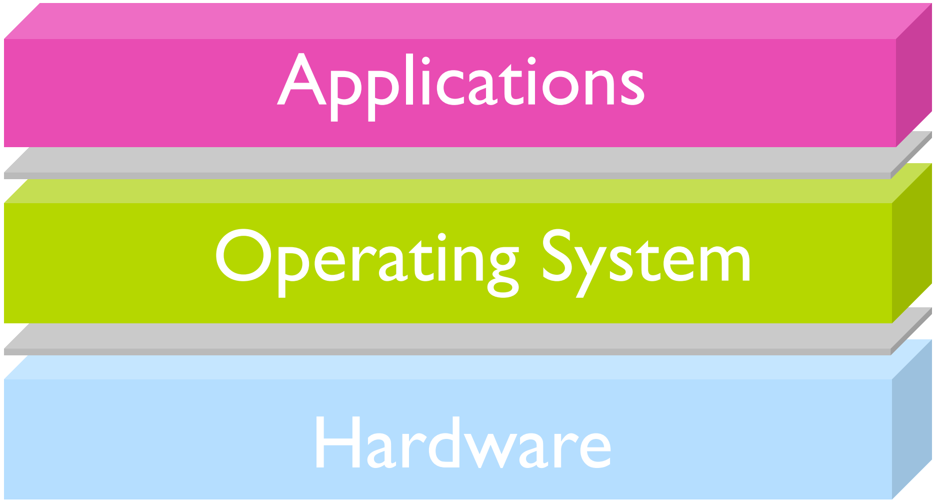
- Impossible to learn only in theory
- You need to get ‘dirty hands’
- Start today!
- Manage the easy stuff to be able to focus on the difficult stuff.
- Best way to learn is to do real programming!

Questions?

Introduction to Operating Systems

What is an Operating System?

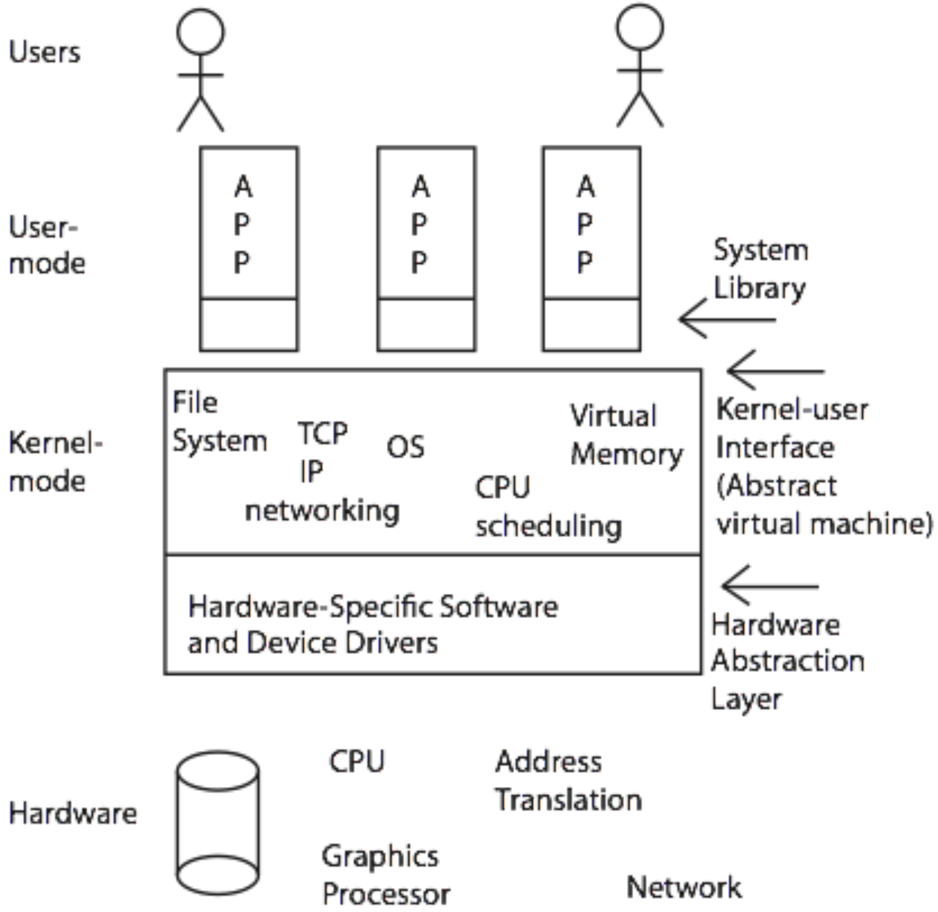
- The operating system is the software layer between user applications and the hardware



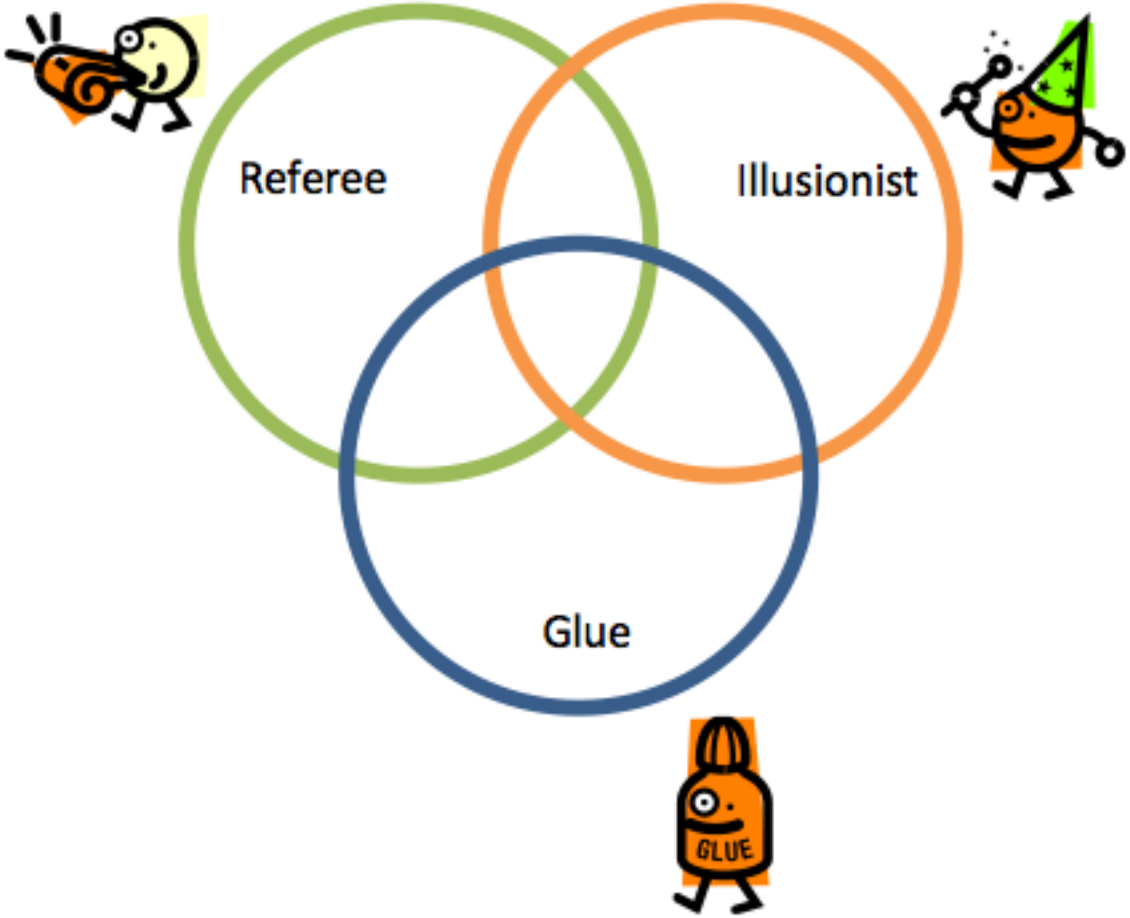
- The OS is "all the code that you didn't have to write" to implement your application

What is an Operating System?

Software to manage a computer's resources for its users and applications.



Operating System Roles

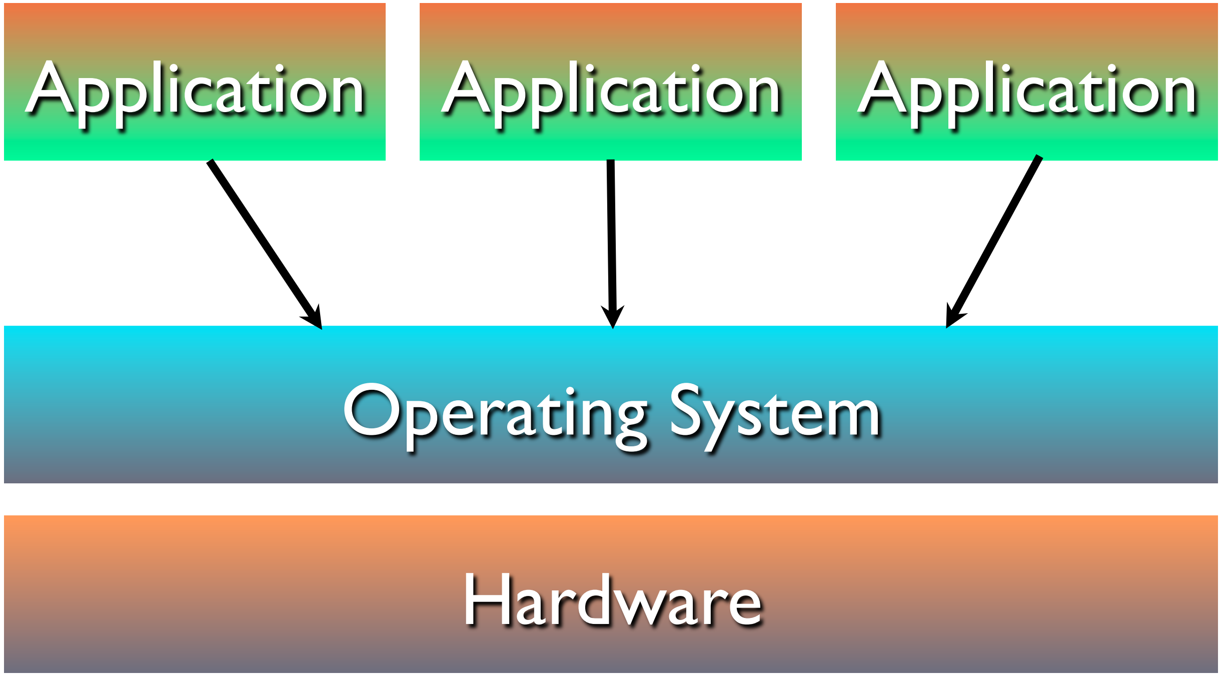


Operating System Roles

- **Referee**
  - Resource allocation: users/applications
  - Isolation of: users/applications from each other
  - Communication between users/applications
- **Illusionist**
  - Applications think they have the entire machine to itself
  - Infinite number of processors and infinite amount of memory
  - Reliable storage and networking
- **Glue**
  - Libraries, user interface widgets, cut/past, same look and feel, abstract away hardware

The Referee

Resource Manager



The OS as Referee

- Sharing
  - Multiplex hardware among applications
    - CPU, memory, devices
  - Applications shouldn’t need to know about each other

The OS as Referee

- Protection
    - Ensure one application can’t R/W another’s data:
      Memory, disk, over network
    - Ensure one application can’t use another’s resources:
      CPU, storage space, bandwidth

The OS as Referee

- Communication
  - Protected applications must still communicate

Resource Management Goals

- Fairness
  - No starvation, every application makes progress
- Efficiency
  - Best use of complete machine resources
  - Minimize, e.g. power consumption
- Predictability
  - Guarantee real-time performance

Resource Management Goals

- Fairness
  - No starvation, every a
- Efficiency
  - Best use of complete machine resources
  - Minimize, e.g. power consumption
- Predictability
  - Guarantee real-time performance

Mutually Contradicting

Example: Threads are Virtualized CPUs

- Threads are virtual CPUs
  - Physical resource: CPUs
  - Virtual resource: Threads
  - Mechanisms: preemption, time slicing, context switching, scheduling
- Much more on this later...

The Illusionist

- Virtualization
- OS creates illusion of a real hardware resource
  - CPU, storage, network

How?

- Multiplexing
  - Divide resources among applications
- Emulation
  - Create the illusion of a hardware resource using software
- Aggregation
  - Join multiple resources to create a larger one

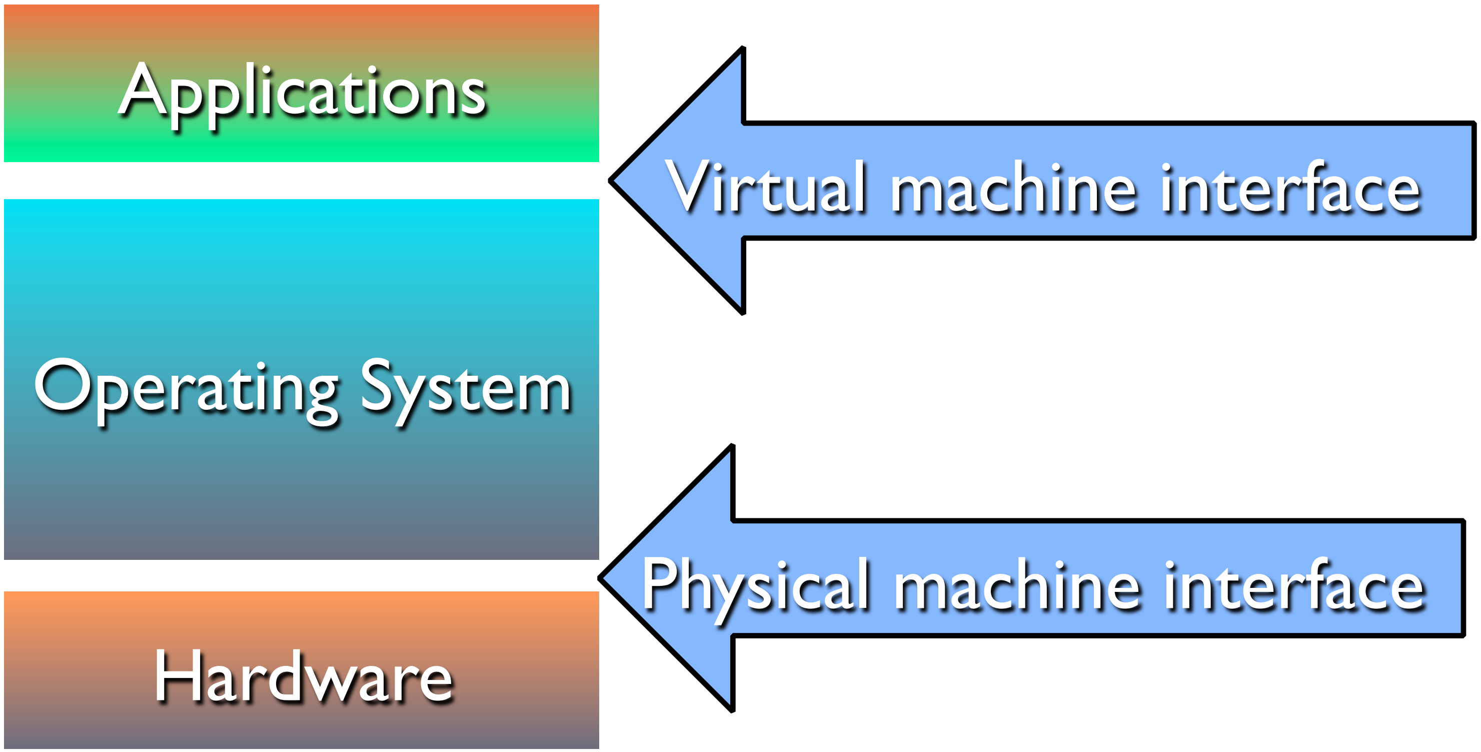
Why?

- Sharing
  - Allow multiple users of a single resource
- Sandboxing
  - Prevent a user from accessing other resources
- Decoupling
  - Avoid tying a user to a particular instance of a resource
- Abstraction
  - Make a resource easier to use

Example:Virtual Memory

- Easier to manage memory
  - Physical resource: RAM
  - Virtual resource:Virtual Memory
  - Method: Multiplexing
  - Mechanism:Virtual Address Translation

The Glue: OS as an Abstract Machine



Applications

Virtual machine interface

Operating System

Physical machine interface

Hardware

The OS as Glue

- Provides high-level abstractions
  - Standardized interfaces
  - Easier to program
  - Shared functionality for all applications
- No direct programming of hardware
  - Hides details of hardware

Services provided by an OS

- Program execution: Load program, execute on CPUs
- Access to I/O devices: Disk, network, keyboard, screen, mouse...
- Protection and access control: For files, connections, etc.

Lots of Operating System Concepts

- System calls
- Concurrency and asynchrony
- Processes and threads
- Security, authorization, protection
- Memory, virtual memory, and paging
- Files and file systems, data management
- I/O: Devices, interrupts, DMA
- Network interfaces and protocol stacks

Operating System Challenges

Operating System Challenges: Reliability

- Does the system do what it was designed to do?
- Does it keep working?
- Potential reliability problems:
  - memory or disk corruption
  - data loss
  - OS failure without data loss
- Reliability metric: time until first failure

Operating System Challenges: Availability

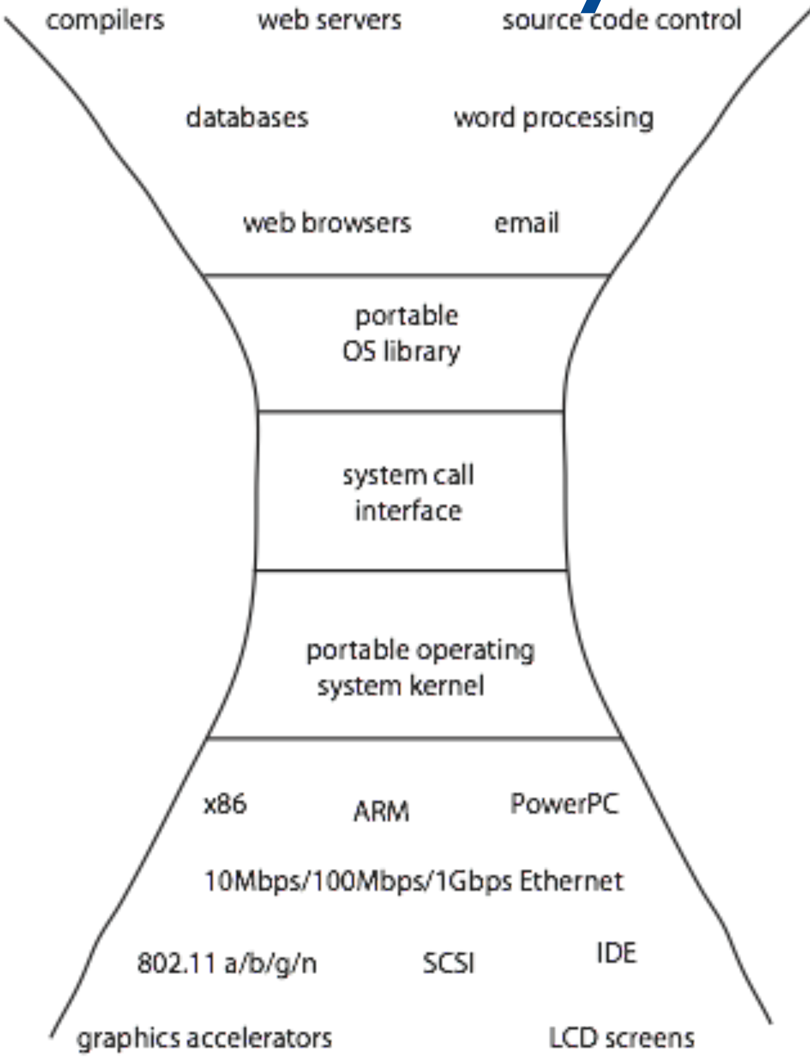
- What portion of the time is the system working?
  - Mean Time To Failure (MTTF)
  - Mean Time To Repair (MTTR)
  - A= MTTF / (MTTF+MTTR)

Operating System Challenges: Security

- Can it be compromised?
  - A faulty application
  - An attacker
- Privacy
  - Data is accessible only to authorized users
- Fault isolation

Operating System Challenges: Portability

- For programs
  - API
  - AMI
- For OS
  - HAL



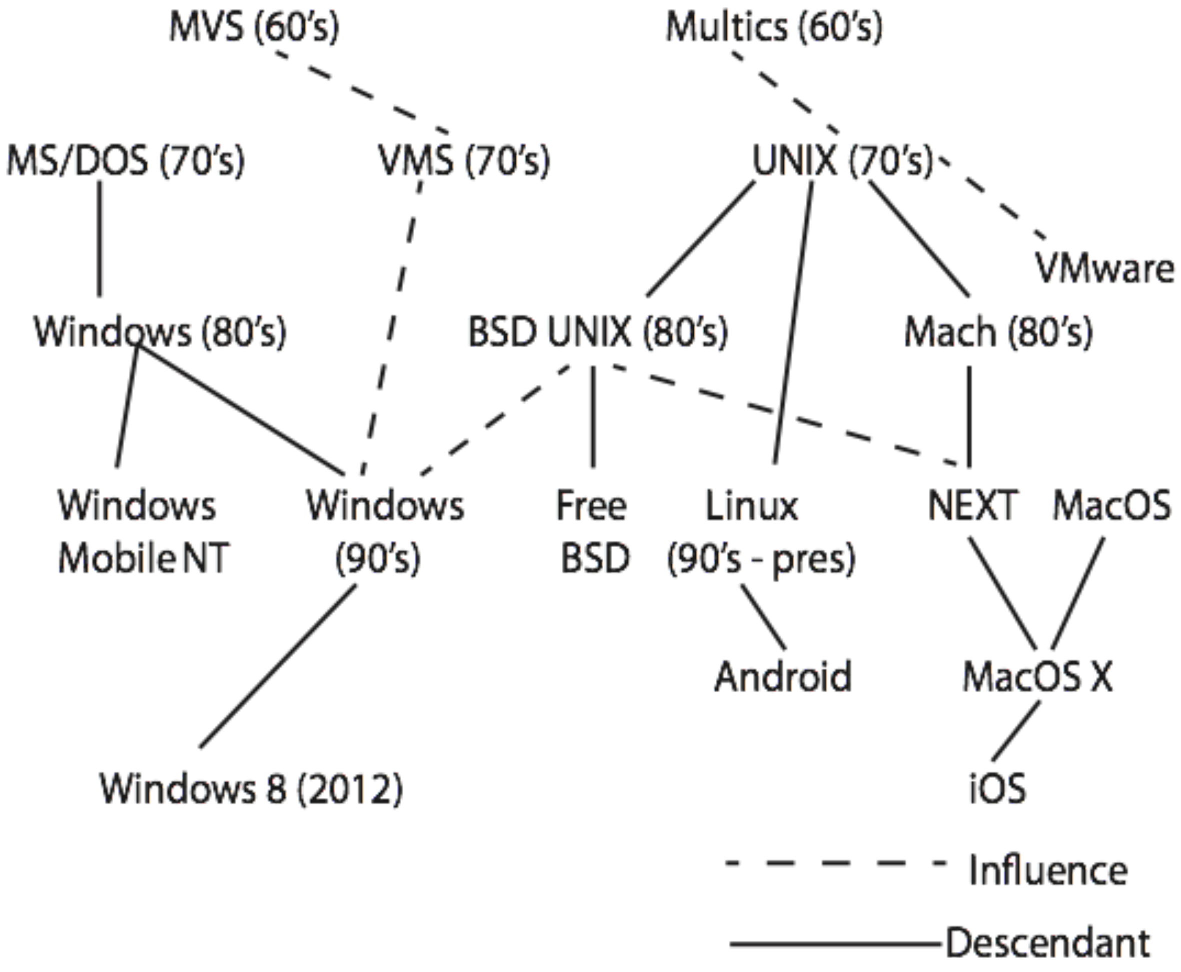
Operating System Challenges: Performance

- **Latency/response time/delay**: How long does an op take to complete?
- **Throughput**: How many ops can be done per unit of time?

Operating System Challenges: Performance

- **Overhead**: How much extra work is done by OS?
- **Fairness**: How equal is the performance of different users?
- **Predictability**: How consistent is the performance over time?

OS History


MVS (60's)
Multics (60's)
MS/DOS (70's)
VMS (70's)
UNIX (70's)
VMware
Windows (80's)
BSD UNIX (80's)
Mach (80's)
Windows Mobile NT
Windows (90's)
Free BSD
Linux (90's - pres)
NEXT
MacOS
Android
MacOS X
Windows 8 (2012)
iOS
Influence
Descendant