CSE451 Introduction to Operating Systems
Spring 2001

Gary Kimura

Lecture #1

March 26, 2001

1

Today

- Class details
- CSE451 educational objectives
- A quick look at what is an operating system

- The class web page should be up and running soon
  - Class information
  - Lecture notes
  - Project and homework assignments
  - Helpful hints and other pointers
  - Email discussion

2

Class details

- Instructor:
  Gary Kimura (GaryKi@cs.washington.edu)
  Office hours: Sieg 419 MWF 1:00 to 2:00 or by appointment
  Email is the best way to contact me
- TAs:
  Jochen Jaeger (JJ@cs.washington.edu)
  Office hours: Sieg 226b Tuesday 1:00 to 2:00
  Adam Prewett (Prewett@cs.washington.edu)
  Office Hours: TBD
- Please ask questions during class and make the lectures interactive

3

Grading (subject to adjustment)

- 20% Homework
  - Approximately 5 to 6 assignments
  - One week from when the assignment is handed out to when it it due
- 40% Exams
  - Two midterms (April 20 and May 11)
  - Final
- 40% Projects
  - Tentatively 4 projects
  - At least two weeks to finish each project

4

Tentative class outline

- OS Overview (chapters 1, 2, and 3)
- Process Management (chapters 4, 5, 6, and 7)
- Storage Management (chapters 8, 9, 10, and 11)
- I/O Systems (chapters 12, 13, and 14)
- Accounting, Protection, and Security (chapters 19 and 20)
- Distributed Systems (chapters 15 and 16)

5

Your job for week #1

- Readings in Silberschatz
  - Chapter 1 (Monday lecture),
  - Chapter 2 (Wednesday lecture)
  - Chapter 3 (Friday lecture)
- Homework #1
  - Out: Today Monday March 26, 2001
  - Due: Next Monday April 2, 2001
  - Silberschatz questions 1.4, 2.5 (justify your answers), 2.9, and 3.6
- Thursday Project #1 will be handed out

6

CSE 451 Education objectives

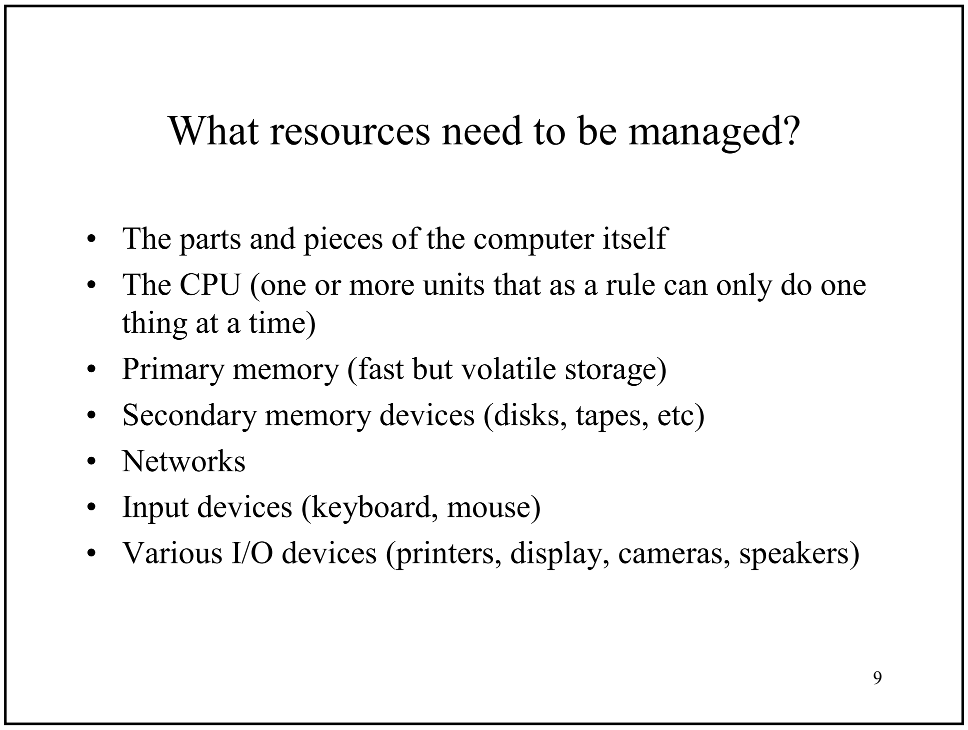
- Two views of an OS
  - The OS user's (i.e., application programmers) view
  - The OS implementer's view
- In this class we will learn:
  - What are the parts of an O.S.
  - How is the O.S. and each sub-part structured
  - What are the important interfaces
  - What are the important policies
  - What algorithms are typically used
- We will do this through reading, lectures, and a project.
- You will need to keep up with all three of these.

7

What is an Operating System?

- The average computer user has trouble understanding this question
- Naively it is "All the code that you didn't write"
- The code that manages physical (hardware) resources
- Provides users (application programmers) with "logical" well-behaved environment
- O.S. defines a set of logical resources (objects) and a set of well-defined operations on those objects (i.e., an interface to use those objects)
- Provides mechanisms and policies for the control of objects/resources
- Controls how different users and programs interact
- Without an O.S. there would be a lot more work to write and run programs

8

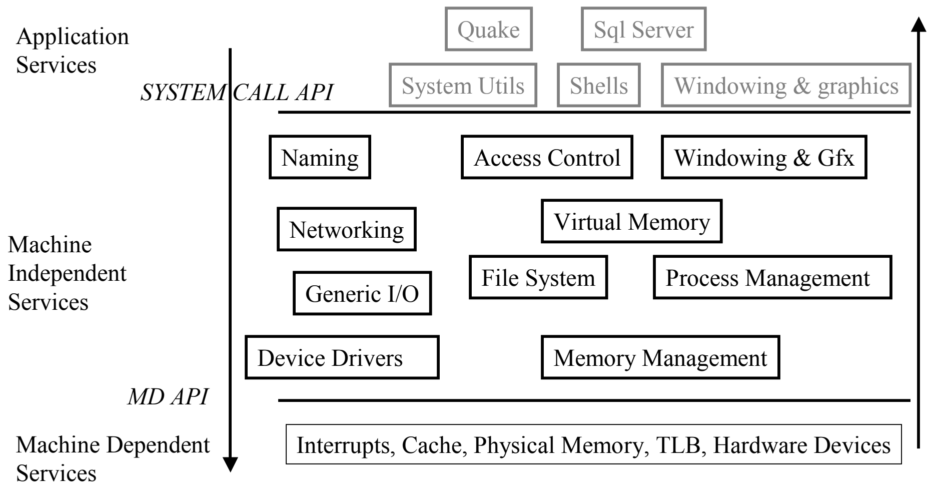


What resources need to be managed?

- The parts and pieces of the computer itself
- The CPU (one or more units that as a rule can only do one thing at a time)
- Primary memory (fast but volatile storage)
- Secondary memory devices (disks, tapes, etc)
- Networks
- Input devices (keyboard, mouse)
- Various I/O devices (printers, display, cameras, speakers)

9

What's in an OS?



Logical OS Structure

10

The OS is Everywhere

```
main(int argc, char **argv)
{ int fd = open(argv[1], O_RDONLY);
   if (fd < 0) {
     fprintf(stderr, “Failed to open\n”);
     exit(-1);
   }
   while (1) {
     if (read(fd, &c, sizeof c) != 1)
       exit(-1);
    putc( c )
  }
}
```

```
% cc main.c

% ./a.out /tmp/foo.bar
```

- Edit
- Compile
- Run/Create Process
- Invoke main
- Open file
  - Check access
  - Cache
  - Read character
- Write character
- Terminate process on EOF or Error

11

Major issues in Operating Systems

- Structure – how is an operating system organized?
- Sharing – how are resources shared among users
- Naming – how are resources named (by users or programs)
- Protection – how is one user/program protected from another
- Security – how to restrict the flow of information
- Performance – why is it so slow?
- Reliability and fault tolerance – when something goes wrong
- Extensibility – how do we add new features?
- Communication – how and with whom can we communicate (exchange information)

12

Major issues in OS (2)

- Concurrency – how are parallel activities created and controlled?
- Scale and growth – what happens as demands or resources increase?
- Persistence – how to make data last longer than programs
- Compatibility – can we ever do anything new?
- Distribution – Accessing the world of information
- Accounting – who pays the bills, and how do we control resource usage?

13

A brief history of operating systems

- “In the beginning”, the OS was just code to which you linked your program, loaded the whole thing into memory, and ran your program; basically, just a run-time library
- Simple batch systems were first real operating systems:
  - O.S. was stored in part of primary memory
  - It loaded a single job (from card reader) into memory
  - Ran that job (printed its output, etc.)
  - Loaded the next job...
  - Control cards in the input file told the O.S. what to do
- Spooling and buffering allowed jobs to be read ahead of time onto tape/disk or into memory.

14

Multiprogramming

- Multiprogramming systems provided increased utilization
  - Keeps multiple runnable jobs loaded in memory
  - Overlaps I/O processing of a job with computes of another
  - Benefits from I/O devices that can operate asynchronously
  - Requires the use of interrupts and DMA
  - Tries to optimize throughput at the cost of response time

15

Timesharing

- Timesharing supported interactive use of
  - Each user feels as if he/she has the entire machine (at least late at night!)
  - Timesharing tries to optimize response time at the cost of throughput
  - Based on time-slicing -- dividing CPU equally among the users
  - Permitted active viewing, editing, debugging, participation of users in the execution process
- MIT Multics system (mid-late 1960s) was first large timesharing system

16

Distributed Operating Systems

- Distributed systems facilitate use of geographically distributed resources
  - Machines connected by wires
- Supports communication between parts of a job or different jobs
  - Interprocess communication
- Sharing of distributed resources, hardware and software
  - Resource utilization and access
- Permits some parallelism, but speedup is not the issue

17

Parallel Operating Systems

- Support parallel applications wishing to get speedup of computationally complex tasks
- Needs basic primitives for dividing one task into multiple parallel activities
- Supports efficient communication between those activities
- Supports synchronization of activities to coordinate sharing of information
- It’s common now to use networks of high-performance PCs/workstations as a parallel computer

18

Embedded Operating Systems

- The decreased cost of processing makes computers ubiquitous. Each "embedded" application needs its own OS or control software:
  - Cell phones
  - PDAs (Palm Pilot, etc.)
  - "Network terminals" (internet interfaces)
- In the future, your house will have 100s of these things in it (if it doesn't already)

19

Future Operating Systems

- Who knows that future is going to bring?
- An OS for a ubiquitous computing environment?
- Radically different programming and usage paradigms will necessitate changes to the OS
- Without a crystal ball it's hard to say where this will end
- This class will stress some of the fundamental parts that will probably always be needed in any basic OS (but there are no guarantees)

20

Next time

- What features in the Hardware do an OS need?

21