Math 4970: Operating Systems

Dr. Bill Slough

Spring 2008

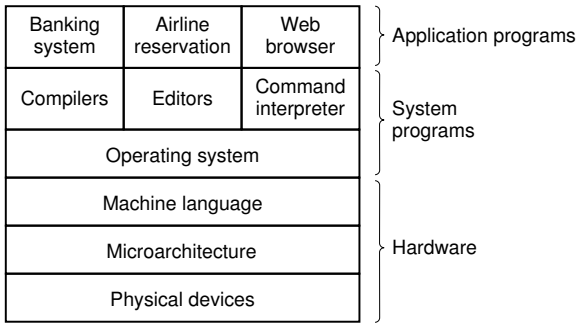
About this course

- Concepts common to all operating systems
  - Processes
  - Deadlocks
  - Memory Management
  - Input/Output
  - File Systems
- Specifics of UNIX
  - Interacting with the system
  - Some common commands
  - Command pipelines
  - Shell scripts
  - System calls
- Specifics of Windows
  - DOS file system

Why Learn About Operating Systems?

- Learn more about computer systems
- Appreciate how OS simplifies our lives as programmers
- Appreciate how OS contributes to efficiency
- Appreciate algorithmic aspects of OS
- Become more sophisticated user of OS

Operating Systems: One Piece of the Puzzle



Hardware is Complex!

Consider floppy disk I/O – NEC PD765 controller chip

- PD 765 has 16 commands
- Each command takes between 1 and 9 bytes
- read requires 13 parameters, packed into 9 bytes:
  - address of the disk block
  - number of sectors per track
  - recording mode used on physical medium
  - intersector gap spacing
  - and more!
- Result: 23 status and error fields, packed in 7 bytes

Software Complexity

Date	Developer	OS	Lines of C	U's
1976	Thompson/Ritchie	AT&T UNIX	9,000	1
1987	Tanenbaum	Minix	62,000	6.9
1991	Torvalds	Linux	1,000,000	111
2000	Cutler *et al.*	Windows NT	28,000,000	3,111

Understanding Large Programs

Print the source code for AT&T Unix:

- Assume 50 lines per page, one side only
- 9000/50 = 180 pages

One person, with study, could understand all of this code.

Understanding Large Programs, Revisited

Print the source code for Windows NT:

- Assume 100 lines per page (50 lines, both sides)
- 28,000,000/100 = 280,000 pages
- 280,000/500 = 560 reams
- 560 × 2 = 1,120 inches
- 1,120/12 = 93.3 feet
- 93.3/3 = 31.1 yards

Can any one person hope to understand this much?

Two Views of An Operating System

- The operating system is a resource manager
- The operating system provides an extended machine

Using hardware resources efficiently
vs.
Thinking of the computer in more abstract ways

Using Hardware Resources Efficiently

Hardware Resources include:

- Processor
- Memory
- Disks
- Network interfaces
- Printers
- Other I/O devices

Thinking of the Computer in More Abstract Ways

Without abstraction, computers would probably be used by relatively few people

Abstractions include:

- Processes
- File systems: hierarchies
- File systems: mechanisms for reading/writing
- Virtual memory
- Network abstractions
- Abstractions for I/O devices

A User Interacting at the Command Line

```
cfwas@OLDM-020070:$ cat Hamlet | wc
```

What happens?

- The program `cat` must run
- The program `wc` must run
- Contents of the file Hamlet must be located and read
- Data produced by **`cat`** is consumed by **`wc`**

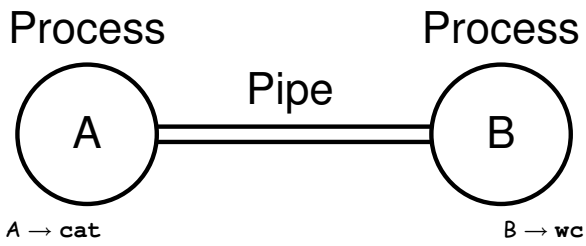
The power of abstraction!

OS Facilities Used By This Pipeline

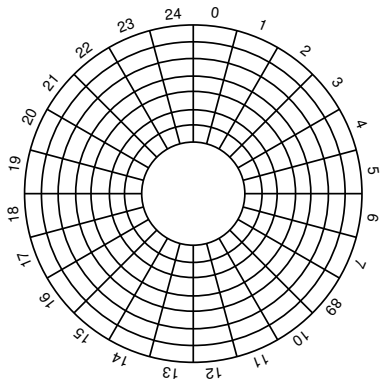
```
cfwas@OLDM-020070:$ cat Hamlet | wc
```

- Process creation and termination
- Multiprogramming
- Data Pipeline
- Synchronization of Processes
- File I/O

Processes and Pipes



Tracks and Sectors



Algorithm and Data Structure Issues

Given the file name Hamlet:

- search the current working directory
- Determine which sectors belong to this file
- Read these sectors, one at a time, in the correct order

Issues for operating system designer to consider:

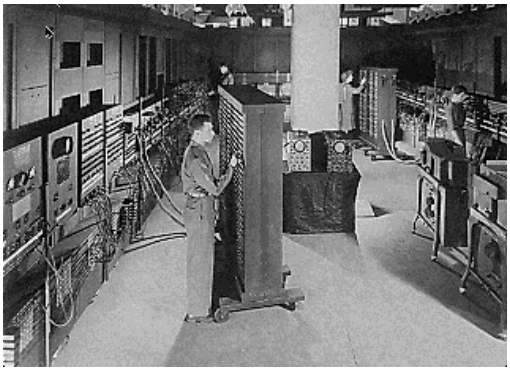
- How is the directory stored?
- What algorithm will be used to search?
- How will the sectors be determined?
- During disk I/O, keep the CPU usefully occupied

Brief History of Operating Systems

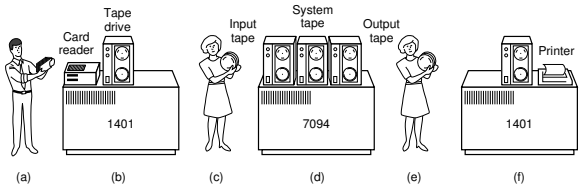
Generation	Era	Hardware	OS
First	1945-1955	Vacuum Tube	None
Second	1955-1965	Transistor	Batch
Third	1965-1980	SSI and MSI	Multiprogramming
Fourth	1980-	LSI and VLSI	Networking, GUI

1st Generation: No Operating System

- No programming languages-machine language only



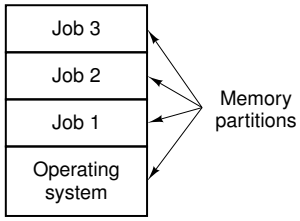
2nd Generation: Batch Systems



- Collect many programs as a first step
- Run these programs sequentially, back-to-back
- I/O is done by relatively cheaper computers
- Goal: keep the expensive computer productively busy!

3rd Generation: Multiprogramming

- Store two or more programs in memory at once
- Provide illusion of concurrent execution
- IBM OS/360
  - 5,000 programmers
  - 5 years
  - Approximately one million lines of code
- Motivation: keep CPU busy during I/O



3rd Generation Innovations

- Multiprogramming: good utilization of CPU
- Timesharing: multiple users, teletypes, one computer
- Implementation: OS is written in high-level language
- Influential Systems: OS/360, CTSS, MULTICS
- Complexity: up to 20 million lines of code for OS

4th Generation: Workstations and PC's

- Workstations
  - High resolution graphics
  - Engineering applications
  - UNIX + X Window System
- Personal Computers
  - Monoprogramming led to multiprogramming
  - Command-line interface led to GUI
  - Examples: CP/M, DOS, Windows, Minix, Linux