Operating Systems
CS451
Spring 1998

Brian Bershad
http://www.cs.washington.edu/homes/bershad
http://www.cs.washington.edu/homes/bershad/cs451

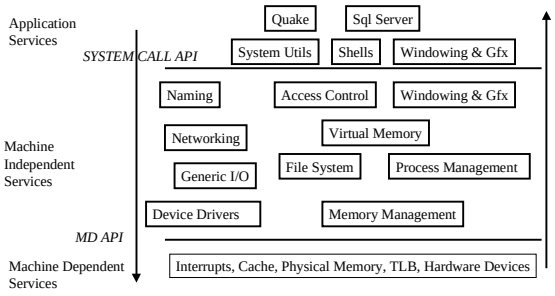
What is an Operating System?

- **"all the code that you didn't write"**
- **the code that manages physical (hardware) resources**
- **provides users 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 the use of those objects)**
- **provides *mechanisms* and *policies* for the control of objects/resources**
- **controls how different users and programs interact**

What resources need to be managed?

- **the CPU (computation)**
- **primary memory**
- **secondary memory devices (disks, tapes, etc)**
- **networks**
- **input devices (keyboard, mouse)**
- **various I/O devices (printers, display, cameras, speakers)**

What's in an OS?



Logical OS Structure

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 )
  }
}
```

- Edit
- Compile
- Run/Create Process
- Invoke main
- Open file
  - check access
  - cache
  - read character
- Write character
- Terminate process on EOF or Err

% cc main.c

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

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)**

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?**

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:**
  - os 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 os what to do
- ***Spooling* and *buffering* allowed jobs to be read ahead of time onto tape/disk or into memory.**

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

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**

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**

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**

CSE 451

- **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.**

Course Outline

- Week 1: 3/30-4/3. History and intro to computer architecture
  - memory, stacks, interrupts, calling conventions, registers, machine state,
- Week 2: 4/6-4/10. Processes
  - process management, scheduling, Unix Fork
- Week 3: 4/13-4/17. Virtual memory
  - MMUs, page tables, paging vs segmentation, replacement policies
- Week 4: 4/20-5/1. Threads and Synchronization
  - Concurrency, thread APIs, Monitors/mutexes/condition variables
- Week 6: 5/4-5/8. Storage Systems
  - Handles, File system APIs, Disks, I/O, Head scheduling, Caching,
- Week 7: 5/11-5/15. Intro to Distributed Systems
  - Moving Bits, Sockets, Remote Procedure Call
- Week 8: 5/18-5/22. Distributed File Systems
  - Concept, Case Studies (NFS, AFS, WWW)
- Week 9: 5/25-5/29. Reliability
  - Transactions, Durability, Recovery
- Week 10: 6/1-6/5. The Internet As a Distributed System
  - DNS, Mobile code, Java