1 Introduction

What is an operating system?

Layer of software between the hardware and application programs. Two main functions:

- Resource manager
- Extended (abstract) machine

OS as resource manager

- mediator/coordinator: resolve conflicting resource demands
- protect users from each others (and from themselves)
- *mechanisms* and *policies* for control of resources, flow of information

OS as extended machine

- provides stable, portable, reliable, safe, well-behaved environment (ideally)
- Magician: makes computer appear to be more than it really is
- Single processor appears like many separate processors
- Single memory made to look like many separate memories, each potentially larger than the real memory

- Abstractions that are easier to program and reason about

Physical machine		Extended machine
Processor(s)		Threads
Memory		Processes
Disks		Files
Network adaptors	Operating System	Comm. channels
Monitor		
Speaker		
Microphone		
Clock/Timer		

What resources need to be managed?

- processor(s) (computation)
- main memory
- secondary memory (disks, tapes, CD-ROM)
- network links
- I/O devices (terminals, printers, audio, video)

OS must

- service all of these devices simultaneously,
- support safe, efficient, and fair sharing of resources among users and programs.

Major issues in operating systems

- concurrency
- sharing
- naming
- protection
- security
- performance
- structure
- reliability and fault tolerance
- extensibility
- communication
- scale and growth
- persistence
- distribution
- accounting

Brief history of operating systems

- In the beginning, one user/program at a time, no overlap of computation and I/O.
- simple *batch systems* were first real OS:
  - OS stored in part of main memory
  - it loaded a single job (from card reader) into memory
  - ran the job, printed its output, etc.
  - loaded next job
- *Spooling* and *buffering*
- *Multiprogramming* systems
  - multiple runnable jobs loaded in memory
  - overlap I/O processing of one job with computation of another
  - benefit from I/O devices that can operate asynchronously
  - requires use of interrupts/DMA
  - tries to optimize *throughput*
- *Timesharing systems*
  - each user feels as if he/she has the entire machine (at least at at night)
  - tries to optimize response time
  - based on time-slicing
  - permits interactive work;
  - MIT Multics system was first large timesharing system (mid-late 1960s)
  - requires periodic clock interrupts
- *Distributed* operating systems
  - facilitate use of geographically distributed resources

- supports communication between parts of a job or different jobs
- sharing of distributed resources, hardware and software
- permits parallelism, but speedup is not necessarily the main objective
- not covered in this course—check out Distributed Systems course

- Characteristics of current OS'es:
  - Large
    - Millions of lines source code
    - 100-10,000 person-years
  - Complex
    - asynchronous
    - concurrent/parallel
    - abundance of hardware platforms with different idiosyncrasies
    - conflicting needs of different application programs and users
    - performance and dependability are crucial

What we'll cover in this course

- Process management
  - Threads and processes
  - Synchronization
  - Multiprogramming
  - CPU Scheduling
  - Deadlock
- Memory management
  - Dynamic storage allocation
  - Sharing main memory
  - Virtual memory
- I/O management
  - File storage management
  - Naming
  - Concurrency
  - Performance
- Advanced topics (based on research papers)
  - Virtual machines
  - Multi-cores (OS structure, scalability)
  - Energy management
  - Distributed systems