COMP 421: Operating Systems and Concurrent Programming

Peter Druschel
Rice University

Spring 2000

1 Introduction

Required readings: Silberschatz/Galvin: Chapters 1-3

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 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

Physical machine		Extended machine
Processor(s)		Threads
Memory		Processes
Disks		Files
Network adaptors	Operating	Comm. channels
Monitor	System	
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 —how are parallel activities created and controlled?
- sharing —how are resources shared among users?
- naming —how are resources named (by programs and users)

- protection —how is one user/program protected from another?
- security —how to restrict the flow of information
- performance —why is it so slow?
- structure —how is an operating system organized?
- reliability and fault tolerance —how to handle failures
- extensibility —how do we add new features?
- communication —how and with whom can we communicate?
- scale and growth —what happens as demands and resources increase?
- persistence —how to make data last longer than programs
- distribution —accessing the world of information
- accounting —who pays the bills, and how do we control resource usage?

Brief history of operating systems

- In the beginning, one user/program at a time, no overlap of computation and I/O. OS first appeared as a subroutine library shared by all users.
- 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* allowed jobs to be read ahead of time onto tape/disk or into memory.
- *Multiprogramming* systems provided increased utilization
  - 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* support interactive use
  - 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 —dividing available CPU time equally among the users
  - permits interactive work; participation of users in the execution process
  - 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 part 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—see COMP 520

- Characteristics of current OS'es:
  - Large
    * 100k's of lines source code
    * 100-1000 person-years
  - Complex
    * asynchronous
    * concurrent/parallel
    * abundance of hardware platforms with different idiosyncrasies
    * conflicting needs of different application programs and users

    * performance is 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
- Protection and Security
- Network Communication
- Operating system structure