Operating Systems : Overview

- Bina Ramamurthy

CSE421

Topics for discussion

- What will you learn in this course? (goals)
- What is an Operating System (OS)?
- Process
- Resources
- System call
- Summary

Goals for the course

- Study the working of an OS.
- Study the design and implementation of various components of an OS.
- Learn about the alternatives available to a designer at all levels of abstraction in an OS.
- Learn concurrent programming using processes, threads, and system calls.
- Understand the basics of distributed systems.
- Explore how you may contribute to solving many open problems in OS and distributed systems.

What is an Operating system?

- Interface manager
  - Human interaction made easy
  - interfacing, abstraction, control and sharing
- Resource manager
  - Efficient use of resources
- Enhances hardware features
  - "virtual" time, space and resource (processes, threads)
- System and data security and protection provider

User Interface

- Operating system provides these facilities for the user:
  - Program creation : editors, debuggers, other development tools.
  - Program execution : load, files, IO operations.
  - Access to IO devices: Read and writes.
  - Controlled access to files: protection mechanisms, abstraction of underlying device.
  - System access: Controls who can access the system.
  - Error detection and response: external, internal, software or hardware error.
  - Accounting: Collect stats., load sharing , for billing purposes.

Resource Manager

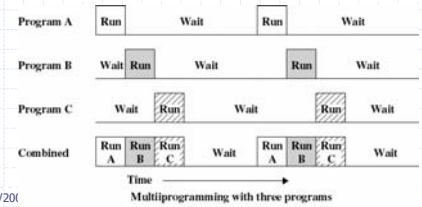
- Processors : Allocation of processes to processors, preemption, scheduling.
- Memory: Allocation of main memory.
- IO devices : when to access io devices, which ones etc.
- Files: Partitions, space allocation and maintenance.
- Applications, Data, objects.

Scheduling and resource management

- Scheduling and resource management combination is an Operations Research (OR) problem.
- Goals : Efficient use of resources, satisfy the service time requested by a process, say, in a real-time system and of course, fairness.
- Short-term and long-term scheduling.
- Queuing is one of the basic operations associated with scheduling. Interrupt is another important concept in the context of scheduling.

Multiprogramming

- If memory can hold several programs, then CPU can switch to another one whenever a program is awaiting for an I/O to complete
- This is multitasking (multiprogramming)



Multiprogramming with three programs

Processes

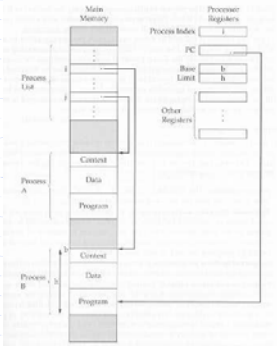
- A program in execution,
- An entity that can be assigned to and executed on a processes,
- It is a unit of work.
- Multiprogramming, time-sharing and real-time transaction systems lead to the refinement of the concept of process.
- A process can be defined by its attributes and behaviors (An OO class definition?).
- When instances of this class co-exist we have concurrent processing.
- Issues in concurrent processing : synchronization, mutual exclusion, deadlock, communication.

Process

- Introduced to obtain a systematic way of monitoring and controlling program execution
- A process is an executable program with:
  - associated data (variables, buffers…)
  - execution context: ie. all the information that
    - the CPU needs to execute the process
      - content of the processor registers
    - the OS needs to manage the process:
      - priority of the process
      - the event (if any) after which the process is waiting
      - Process resources

A simple implementation of processes

- The process index register contains the index into the process list of the currently executing process (B)
- A process switch from B to A consist of storing (in memory) B's context and loading (in CPU registers) A's context



Deadlock

(a) A potential deadlock. (b) an actual deadlock.

Memory management

- Requirements: Process isolation, automatic allocation and maintenance, protection and access control, long-term storage facilities.
- Virtual memory and file system facilities together satisfy all these requirements.
- Virtual memory allows programs to address the memory from a logical point of view without regard to the amount of main memory available.
- File : persistent storage for programs and data.
- Can view file also as an class? File concept makes makes access control and protection convenient for the OS.

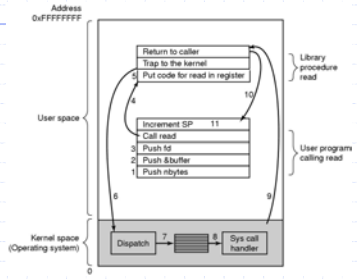
Protection and Security

- When sharing resources, protection of the systems and user resources from intentional as well as inadvertent misuse.
- Protection generally deals with access control. Ex: Read only file
- Security deals usually with threats from outside the system that affects the integrity and availability of the system and information with the system.
- Example: username, password to access system. Data encryption to protect information.

System Structure

- Because of it's enormous complexity, we view the OS system as a series of levels
- Each level performs a related subset of functions
- Each level relies on the next lower level to perform more primitive functions
- Well defined interfaces: one level can be modified without affecting other levels
- This decomposes a problem into a number of more manageable sub problems

System Call



There are 11 steps in making the system call
read (fd, buffer, nbytes)

Some System Calls For Process Management and File Management

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

Operating system Modular View