Operating Systems: Overview

Bina Ramamurthy

CSE421
Topics for discussion

- What will you learn in this course? (goals)
- What is an Operating System (OS)?
- Evolution of OS
- Important OS Components
- Major achievements
- Operating system design hierarchy
- Sample systems
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**.
Multiprogramming

From uniprogramming to multiprogramming systems:

Multiprogramming systems: batch programs, objective: maximize system (processor) utilization.

Time sharing systems: Objective is minimize response time. Typical programs are interactive.
Multiprogrammed Batch Systems

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).

```
<table>
<thead>
<tr>
<th>Program A</th>
<th>Run</th>
<th>Wait</th>
<th>Run</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program B</td>
<td>Wait</td>
<td>Run</td>
<td>Wait</td>
<td>Run</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program C</td>
<td>Wait</td>
<td>Run</td>
<td>Wait</td>
<td>Run</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Combined</td>
<td>Run A</td>
<td>Run B</td>
<td>Run C</td>
<td>Wait</td>
</tr>
</tbody>
</table>

Time
Multiiprogramming with three programs

8/29/2006 B.Ramamurthy
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: it can be viewed as an Abstract Data Type (ADT).
- When instances of this ADT 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
    - other data (that we will introduce later)
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.
- A data structure that provides flexibility (to add new features).
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 ADT? File concept 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.
Scheduling and resource management

- Scheduling and resource management 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.
Scheduling and Resource Management

- **Differential responsiveness**
  - discriminate between different classes of jobs

- **Fairness**
  - give equal and fair access to all processes of the same class

- **Efficiency**
  - maximize throughput, minimize response time, and accommodate as many users as possible
File System

- Implements long-term store (often on disk)
- Information stored in named objects called files
  - a convenient unit of access and protection for OS
- Files (and portions) may be copied into virtual memory for manipulation by programs
System Structure

- Because of its 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.
Structure of OS

Client-Server Model

- **SERVERS**: Splitting the OS into parts, each of which handles one facet of the system, such as file service, process service, terminal service, or memory service

- **CLIENTS**: User processes: A client process obtains services by sending messages to the servers.

Advantages:
- Modularity: A bug in files server will crash only the files server and not the whole OS
- **Adaptability to distributed system**: Services could be provided from a remote computer.
Characteristics of Modern Operating Systems

- New design elements were introduced recently
- In response to new hardware development
  - multiprocessor machines
  - high-speed networks
  - faster processors and larger memory
- In response to new software needs
  - multimedia applications
  - Internet and Web access
  - Client/Server applications
Microkernel architecture

- Only a few essential functions in the kernel
  - primitive memory management (address space)
  - Interprocess communication (IPC)
  - basic scheduling
- Other OS services are provided by processes running in user mode (servers)
  - device drivers, file system, virtual memory...
- More flexibility, extensibility, portability...
- A performance penalty by replacing service calls with message exchanges between process...
Multithreading

- A process is a collection of one or more threads that can run simultaneously.
- Useful when the application consists of several tasks that do not need to be serialized.
- Gives the programmer a greater control over the timing of application-related events.
- All threads within the same process share the same data and resources and a part of the process’s execution context.
- It is easier to create or destroy a thread or switch among threads (of the same process) than to do these with processes.
Symmetric Multiprocessing (SMP)

- A computer with multiple processors
- Each processor can perform the same functions and share same main memory and I/O facilities (symmetric)
- The OS schedule processes/threads across all the processors (real parallelism)
- Existence of multiple processors is transparent to the user.
- Incremental growth: just add another CPU!
- Robustness: a single CPU failure does not halt the system, only the performance is reduced.
Operating system Modular View

- Hardware + interrupts
- Process + primitives
- Devices

- Virtual Mem
- Shell
- Directories

- Comm. Prmtvs
- File sys.

- Application Server
- Application Clients
- Web Server
- Web Clients
- User Processes

8/29/2006 B. Ramamurthy 23
Types of OS

- Multiprocessing - multiple CPUs
- Multiprogramming - Time sharing, interactive
- Real-time: deadlines, time constraints, predictability
- Distributed systems: Sharing and fault tolerance, reliability, dependability.
- Network OS
- Network Transparent Systems: CORBA-like
- Network-centric Systems: Jini-like
- Component-based systems: Enterprise Java

Read Chapter 1