Introduction

- The fundamental task of any operating system is process management.
- OS must allocate resources to processes, enable sharing of information, protect resources, and enable synchronization among processes.
- In many modern OS the problems of process management is compounded by introduction of threads.
- We will process management in this lecture and threads in the next.

Topics for discussion

- Requirement of process
- Process states
- Creation, termination and suspension
- Five State Model
- Process Control Block (PCB)
- Process control
- Unix System V
- Summary

Major requirements

- OS must interleave the execution of a number of processes to maximize processor use while providing reasonable response time.
- OS must allocate resources to processes in conformance with a specific policy. Example: (i) higher priority, (ii) avoid deadlock.
- Support user creation of processes and IPC both of which may aid in the structuring of applications.

Process creation

- Four common events that lead to a process creation are:
  1) When a new batch-job is presented for execution.
  2) When an interactive user logs in.
  3) When OS needs to perform an operation (usually IO) on behalf of a user process, concurrently with that process.
  4) To exploit parallelism an user process can spawn a number of processes.

=> concept of parent and child processes.
Process Hierarchies

- Parent creates a child process, child processes can create its own process
- Forms a hierarchy
  - UNIX calls this a "process group"
- Windows has no concept of process hierarchy
  - all processes are created equal

Termination of a process

- Normal completion, time limit exceeded, memory unavailable
- Bounds violation, protection error, arithmetic error, invalid instruction
- IO failure, Operator intervention, parent termination, parent request
- A number of other conditions are possible.
- Segmentation fault: usually happens when you try write/read into/from a non-existent array/structure/object component. Or access a pointer to a dynamic data before creating it. (new etc.)
- Bus error: Related to function call and return. You have messed up the stack where the return address or parameters are stored

A five-state process model

- Five states: New, Ready, Running, Blocked, Exit
- New: A process has been created but has not yet been admitted to the pool of executable processes.
- Ready: Processes that are prepared to run if given an opportunity. That is, they are not waiting on anything except the CPU availability.
- Running: The process that is currently being executed. (Assume single processor for simplicity.)
- Blocked: A process that cannot execute until a specified event such as an IO completion occurs.
- Exit: A process that has been released by OS either after normal termination or after abnormal termination (error)

State Transition Diagram

- Think of the conditions under which state transitions may take place.

Queuing model

Process Transitions

- Ready --> Running
  - When it is time, the dispatcher selects a new process to run
- Running --> Ready
  - the running process has expired his time slot
  - the running process gets interrupted because a higher priority process is in the ready state
Process Transitions

- **Running --> Blocked**
  - When a process requests something for which it must wait
    - a service that the OS is not ready to perform
    - an access to a resource not yet available
    - initiates I/O and must wait for the result
    - waiting for a process to provide input (IPC)

- **Blocked --> Ready**
  - When the event for which it was waiting occurs

Process suspension

- Many OS are built around (Ready, Running, Blocked) states. But there is one more state that may aid in the operation of an OS - **suspended** state.
- When none of the processes occupying the main memory is in a Ready state, OS swaps one of the blocked processes out onto the Suspend queue.
- When a Suspended process is ready to run it moves into "Ready, Suspend" queue. Thus we have two more state: Blocked_Suspend, Ready_Suspend.

Process suspension (contd.)

- **Blocked_suspend**: The process is in the secondary memory and awaiting an event.
- **Ready_suspend**: The process is in the secondary memory but is available for execution as soon as it is loaded into the main memory.
- State transition diagram on the next slide.
- Observe on what condition does a state transition take place? What are the possible state transitions?

State Transition Diagram (take 2)

- Think of the conditions under which state transitions may take place.

Implementation of Processes

- Skeleton of what lowest level of OS does when an interrupt occurs

Operating System Control Structures

- An OS maintains the following tables for managing processes and resources:
  - Memory tables (see later)
  - I/O tables (see later)
  - File tables (see later)
  - Process tables (this chapter)
Process description

- OS constructs and maintains tables of information about each entity that it is managing: memory tables, IO tables, file tables, process tables.
- **Process control block**: Associated with each process are a number of attributes used by OS for process control. This collection is known as **PCB**.
- For more details on PCB see your text.

Process Table Entry (PCB)

<table>
<thead>
<tr>
<th>Fields of a process table entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process management</td>
</tr>
<tr>
<td>Registers</td>
</tr>
<tr>
<td>Program counter</td>
</tr>
<tr>
<td>Program status word</td>
</tr>
<tr>
<td>Stack pointer</td>
</tr>
<tr>
<td>Process state</td>
</tr>
<tr>
<td>Priority</td>
</tr>
<tr>
<td>Scheduling parameters</td>
</tr>
<tr>
<td>Process ID</td>
</tr>
<tr>
<td>Parent process</td>
</tr>
<tr>
<td>Process group</td>
</tr>
<tr>
<td>Signals</td>
</tr>
<tr>
<td>Time when process started</td>
</tr>
<tr>
<td>CPU time used</td>
</tr>
<tr>
<td>Child's CPU time</td>
</tr>
<tr>
<td>Time at next alarm</td>
</tr>
</tbody>
</table>

Fields of a process table entry

Process control block

- Contains three categories of information:
  1) Process identification
  2) Process state information
  3) Process control information
- **Process identification**:
  - numeric identifier for the process (pid)
  - identifier of the parent (ppid)
  - user identifier (uid) - id of the user responsible for the process.

Process control block (contd.)

- **Process state information**:
  - User visible registers
  - Control and status registers: PC, IR, PSW, interrupt related bits, execution mode.
  - Stack pointers

Process control block (contd.)

- **Process control information**:
  - Scheduling and state information: Process state, priority, scheduling-related info., event awaited.
  - Data structuring: pointers to other processes (PCBs): belong to the same queue, parent of process, child of process or some other relationship.
  - Interprocess comm: Various flags, signals, messages may be maintained in PCBs.

Process control block (contd.)

- **Process control information (contd.)**:
  - Process privileges: access privileges to certain memory area, critical structures etc.
  - Memory management: pointer to the various memory management data structures.
  - Resource ownership: Pointer to resources such as opened files. Info may be used by scheduler.
  - PCBs need to be protected from inadvertent destruction by any routine. So protection of PCBs is a critical issue in the design of an OS.
OS Functions related to Processes

- Process management: Process creation, termination, scheduling, dispatching, switching, synchronization, IPC support, management of PCBs
- Memory management: Allocation of address space to processes, swapping, page and segment management.
- IO management: Buffer management, allocation of IO channels and devices to processes.
- Support functions: Interrupt handling, accounting, monitoring.

Modes of execution

- Two modes: user mode and a privileged mode called the kernel mode.
- Why? It is necessary to protect the OS and key OS tables such as PCBs from interference by user programs.
- In the kernel mode, the software has complete control of the processor and all its hardware.
- When a user makes a system call or when an interrupt transfers control to a system routine, an instruction to change mode is executed. This mode change will result in an error unless permitted by OS.

Summary

- A process is a unit of work for the Operating System.
- Implementation of the process model deals with process description structures and process control methods.
- Process management is the of the operating system requiring a range of functionality from interrupt handling to IO management.