Basic Operating System Concepts

Ram Sarkar, CSE, JU

Main Goals of OS

1. Resource Management: Disk, CPU cycles, etc. must be managed efficiently to maximize overall system performance
2. Resource Abstraction: Software interface to simplify use of hardware resources
3. Virtualization: Supports resource sharing – gives each process the appearance of an unshared resource

Execution Modes (Dual Mode Execution)

- User mode vs. kernel (or supervisor) mode
- Protection mechanism: critical operations (e.g. direct device access, disabling interrupts) can only be performed by the OS while executing in kernel mode
- Mode bit
- Privileged instructions

Topics

- Processes &Threads
- Scheduling
- Synchronization
- Memory Management
- File and I/O Management

Process Definition

- A process is an instance of a program in execution.
- It encompasses the static concept of program and the dynamic aspect of execution.
- As the process runs, its context (state) changes – register contents, memory contents, etc., are modified by execution

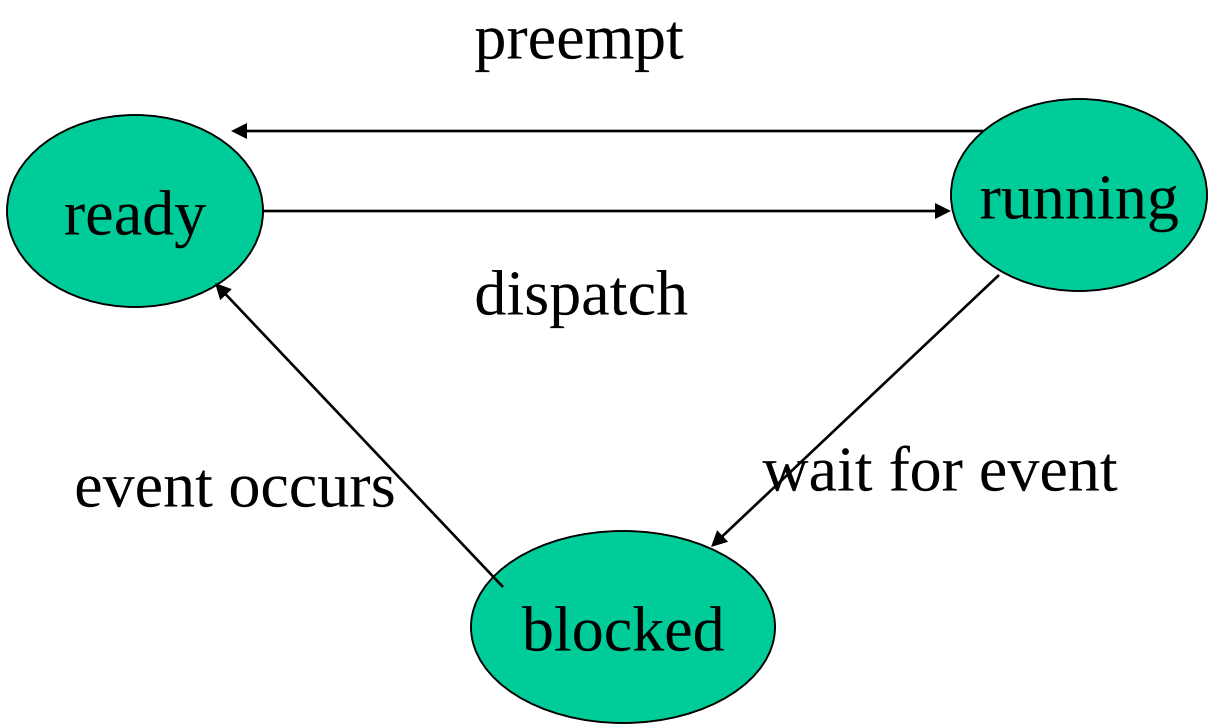
Processes: Process Image

- The process image represents the current status of the process
- It consists of (among other things)
  - Executable code
  - Static data area
  - Stack & heap area
  - Process Control Block (PCB): data structure used to represent execution context, or state
  - Other information needed to manage process

Process Execution States

- A process as being in one of several basic states.
  - Running
  - Ready
  - Blocked (or sleeping)

Process State Transition Diagram



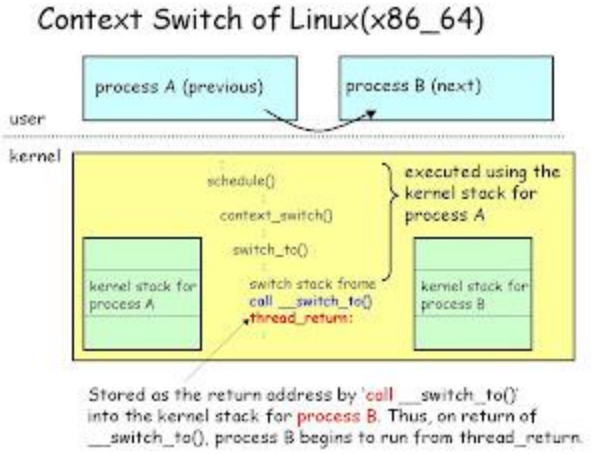
Other States

- New
- Exit
- Suspended (Swapped)
  - Suspended blocked
  - Suspended ready

Context Switch

(sometimes called process switch)

- It involves two processes:
  - One leaves the Running state
  - Another enters the Running state
- The status (context) of one process is saved; the status of the second process restored.
- Don’t confuse with mode switch.



Concurrent Processes

- Two processes are concurrent if their executions overlap in time.
- In a uniprocessor environment, multiprogramming provides concurrency.
- In a multiprocessor, true parallel execution can occur.

Protection

- When multiple processes (or threads) exist at the same time, and execute concurrently, the OS must protect them from mutual interference.
- Memory protection (memory isolation) prevents one process from accessing the physical address space of another process.
- Base/limit registers, virtual memory are techniques to achieve memory protection.

Processes and Threads

- Traditional processes could only do one thing at a time – they were single-threaded.
- Multithreaded processes can (conceptually) do several things at once – they have multiple threads.
- A thread is an “execution context” or “separately schedulable” entity.

Threads versus Processes

- If two *processes* want to access shared data structures, the OS must be involved.
  - Overhead: system calls, mode switches, context switches, extra execution time.
- Two threads in a single process can share global data automatically – as easily as two functions in a single process.

Review Topics

- Processes &Threads
- Scheduling
- Synchronization
- Memory Management
- File and I/O Management

Process Scheduling

- Process scheduling decides which process to dispatch (to the **Run** state) next.
- In a multiprogrammed system several processes compete for a single processor
- **Preemptive scheduling**: a process can be removed from the **Run** state before it completes or blocks (timer expires or higher priority process enters **Ready** state).

Scheduling Goals

- Optimize turnaround time and/or response time
- Optimize throughput
- Avoid starvation (be “fair”)
- Respect priorities
  - Static
  - Dynamic

Review Topics

- Processes &Threads
- Scheduling
- Synchronization
- Memory Management
- File and I/O Management

Interprocess Communication (IPC)

- Processes that cooperate to solve problems must exchange information.
- Two approaches:
  - Shared memory
  - Message passing (copying information from one process address space to another)

- Shared memory is more efficient (no copying), but isn't always possible.

Process Synchronization

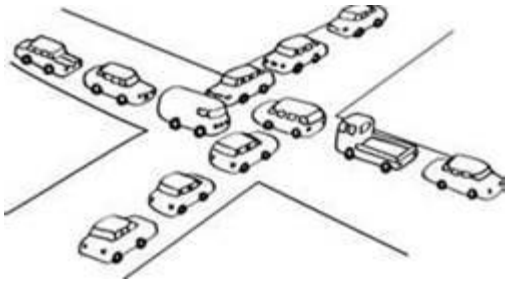
- Concurrent processes are *asynchronous:* the relative order of events within the two processes cannot be predicted in advance.
- If processes are related (exchange information in some way) it may be necessary to synchronize their activity at some points.

Mutual Exclusion

- A critical section is the code that accesses shared data or resources.
- A solution to the critical section problem must ensure that *only one process at a time can execute its critical section (CS).*
- Two separate shared resources can be accessed concurrently.

Deadlock

- A set of processes is deadlocked when each is in the Blocked state because it is waiting for a resource that is allocated to one of the others.
- Deadlocks can only be resolved by agents outside of the deadlock



Deadlock versus Starvation

- Starvation occurs when a process is repeatedly denied access to a resource even though the resource becomes available.
- Deadlocked processes are permanently blocked but starving processes may eventually get the resource being requested.
- In starvation, the resource being waited for is continually in use, while in deadlock it is not being used because it is assigned to a blocked process.

Causes of Deadlock

- Mutual exclusion (exclusive access)
- Wait while hold (hold and wait)
- No preemption
- Circular wait

Review Topics

- Processes &Threads
- Scheduling
- Synchronization
- Memory Management
- File and I/O Management

Memory Management

- Introduction
- Allocation methods
  - One process at a time
  - Multiple processes, contiguous allocation
  - Multiple processes, virtual memory

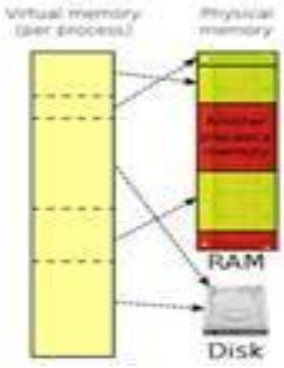
Memory Management - Intro

- Primary memory must be shared between the OS and user processes.
- OS must protect itself from users, and one user from another.
- OS must also manage the sharing of physical memory so that processes are able to execute with reasonable efficiency.

Allocation Methods:

Multiple Processes, Contiguous Allocation

- Several processes resided in memory at one time (multiprogramming).
- The entire process image for each process was stored in a contiguous set of locations.
- Drawbacks:
  - Limited number of processes at one time
  - Fragmentation of memory



Virtual Memory - Paging

- The address space of a program is divided into “pages” – a set of contiguous locations.
- Page size is a power of 2; typically at least 4K.
- Memory is divided into page frames of same size.
- Any “page” in a program can be loaded into any “frame” in memory, so no space is wasted.

Paging - continued

- General idea – save space by loading only those pages that a program needs now.
- Result – more programs can be in memory at any given time
- Problems:
  - How to tell what's "needed"
  - How to keep track of where the pages are
  - How to translate virtual addresses to physical

Solutions to Paging Problems

- How to tell what's "needed"
  - Demand paging
- How to keep track of where the pages are
  - The page table
- How to translate virtual addresses to physical
  - MMU (memory management unit) uses logical addresses and page table data to form actual physical addresses. All done in hardware.

OS Responsibilities in Paged Virtual Memory

- Maintain page tables
- Manage page replacement

Review Topics

- Processes &Threads
- Scheduling
- Synchronization
- Memory Management
- File and I/O Management

File Systems

- Maintaining a shared file system is a major job for the operating system.
- Single user systems require protection against loss, efficient look-up service, etc.
- Multiple user systems also need to provide access control.

File Systems – Disk Management

- The file system is also responsible for allocating disk space and keeping track of where files are located.
- Disk storage management has many of the problems main memory management has, including fragmentation issues.

End of OS Review