Process Description and Control

Chapter 3

All multiprogramming OS are build around the concept of processes

A process is sometimes called a task

OS Requirements for Processes

- **OS must interleave the execution of several processes to maximize CPU usage while providing reasonable response time**
- **OS must allocate resources to processes while avoiding deadlock**
- **OS must support inter process communication and user creation of processes**

Dispatcher (short-term scheduler)

- Is an OS program that moves the processor from one process to another
- It prevents a single process from monopolizing processor time
- It decides who goes next according to a scheduling algorithm (to be discussed later)
- The CPU will always execute instructions from the dispatcher while switching from process A to process B

When does a process gets created?

- Submission of a batch job
- User logs on
- Created by OS to provide a service to a user (ex: printing a file)
- Spawned by an existing process
  - a user program can dictate the creation of a number of processes

When does a process gets terminated?

- Batch job issues *Halt* instruction
- User logs off
- Process executes a service request to terminate
- Error and fault conditions

Reasons for Process Termination

- Normal completion
- Time limit exceeded
- Memory unavailable
- Memory bounds violation
- Protection error
  - example: write to read-only file
- Arithmetic error
- Time overrun
  - process waited longer than a specified maximum for an event

Reasons for Process Termination

- **I/O failure**
- **Invalid instruction**
  - happens when try to execute data
- **Privileged instruction**
- **Operating system intervention**
  - such as when deadlock occurs
- **Parent request to terminate one offspring**
- **Parent terminates so child processes terminate**

Process States

- **Let us start with these states:**
  - The Running state
    - The process that gets executed (single CPU)
  - The Ready state
    - any process that is ready to be executed
  - The Blocked state
    - when a process cannot execute until some event occurs (ex: the completion of an I/O)

Other Useful States

- **The New state**
  - ◆ OS has performed the necessary actions to create the process
    - ☞ has created a process identifier
    - ☞ has created tables needed to manage the process
  - ◆ but has not yet committed to execute the process (not yet admitted)
    - ☞ because resources are limited

Other Useful States

- **The Exit state**
  - ◆ Termination moves the process to this state
  - ◆ It is no longer eligible for execution
  - ◆ Tables and other info are temporarily preserved for auxiliary program
    - ☞ Ex: accounting program that cumulates resource usage for billing the users
- **The process (and its tables) gets deleted when the data is no more needed**

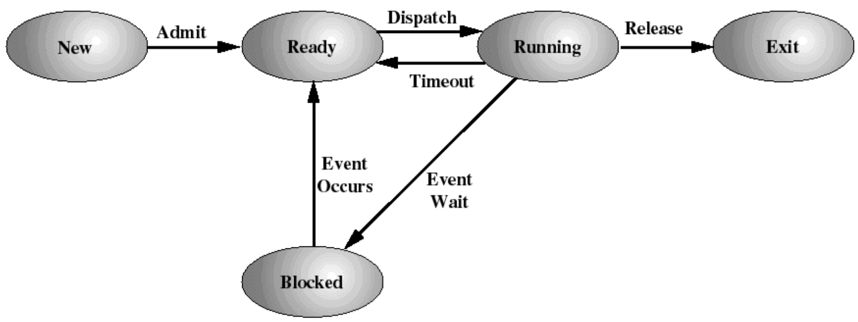
Process Transitions

- **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

A Five-state Process Model



Ready to exit: A parent may terminate a child process

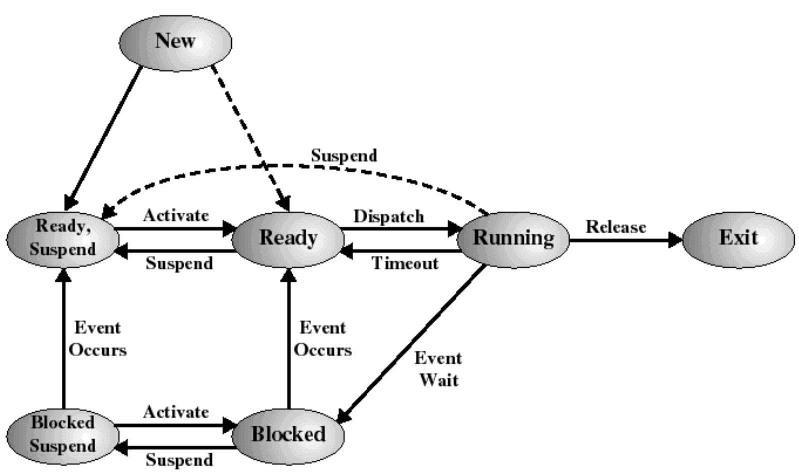
The need for swapping

- So far, all the processes had to be (at least partly) in main memory
- Even with virtual memory, keeping too many processes in main memory will deteriorate the system's performance
- The OS may need to suspend some processes, ie: to swap them out to disk. We add 2 new states:
- Blocked Suspend: blocked processes which have been swapped out to disk
- Ready Suspend: ready processes which have been swapped out to disk

New state transitions (mid-term scheduling)

- **Blocked --> Blocked Suspend**
  - When all processes are blocked, the OS will make room to bring a ready process in memory
- **Blocked Suspend --> Ready Suspend**
  - When the event for which it as been waiting occurs (state info is available to OS)
- **Ready Suspend --> Ready**
  - when no more ready process in main memory
- **Ready--> Ready Suspend (unlikely)**
  - When there are no blocked processes and must free memory for adequate performance

A Seven-state Process Model



Operating System Control Structures

- **An OS maintains the following tables for managing processes and resources:**
  - ◆ Memory tables (to be discussed later)
  - ◆ I/O tables (to be discussed later)
  - ◆ File tables (to be discussed later)
  - ◆ Process tables

Process Image

- **User program**
- **User data**
- **Stack(s)**
  - ◆ for procedure calls and parameter passing
- **Process Control Block (execution context)**
  - ◆ Data needed (process attributes) by the OS to control the process. This includes:
    - ☞ Process identification information
    - ☞ Processor state information
    - ☞ Process control information

Location of the Process Image

- **Each process image is in virtual memory**
  - may not occupy a contiguous range of addresses (depends on the memory management scheme used)
  - both a private and shared memory address space is used
- **The location if each process image is pointed to by an entry in the Primary Process Table**
- **For the OS to manage the process, at least part of its image must be loaded into main memory**

Process Identification (in the PCB)

- **A few numeric identifiers may be used**
  - Unique process identifier (always)
    - indexes (directly or indirectly) into the primary process table
  - User identifier
    - the user who is responsible for the job
  - Identifier of the process that created this process

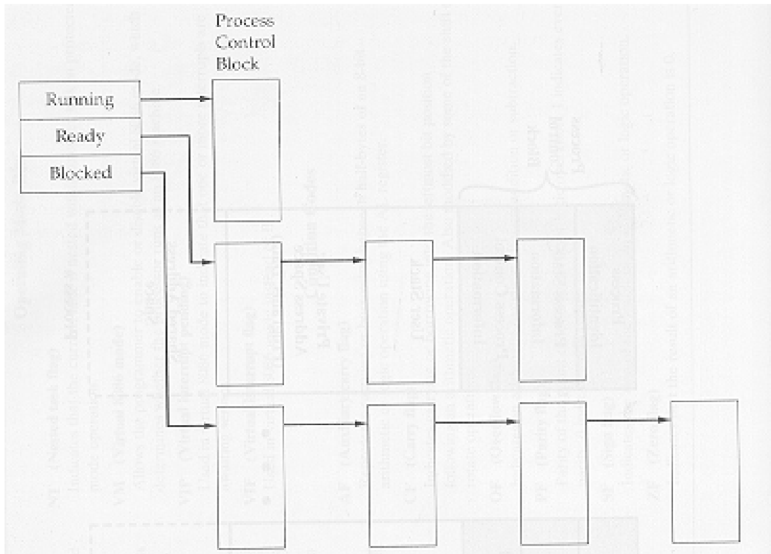
Processor State Information (in PCB)

- **Contents of processor registers**
  - User-visible registers
  - Control and status registers
  - Stack pointers
- **Program status word (PSW)**
  - contains status information
  - Example: the EFLAGS register on Pentium machines

Process Control Information (in PCB)

- **scheduling and state information**
  - Process state (ie: running, ready, blocked...)
  - Priority of the process
  - Event for which the process is waiting (if blocked)
- **data structuring information**
  - may hold pointers to other PCBs for process queues, parent-child relationships and other structures

Queues as linked lists of PCBs



Process Control Information (in PCB)

- **interprocess communication**
  - ◆ may hold flags and signals for IPC
- **process privileges**
  - ◆ Ex: access to certain memory locations...
- **memory management**
  - ◆ pointers to segment/page tables assigned to this process
- **resource ownership and utilization**
  - ◆ resource in use: open files, I/O devices...
  - ◆ history of usage (of CPU time, I/O...)

Modes of Execution

- **To provide protection to PCBs (and other OS data) most processors support at least 2 execution modes:**
  - Privileged mode (a.k.a. system mode, kernel mode, supervisor mode, control mode)
    - manipulating control registers, primitive I/O instructions, memory management...
  - User mode
- **For this the CPU provides a (or a few) mode bit which may only be set by an interrupt or trap or OS call**

Process Creation

- **Assign a unique process identifier**
- **Allocate space for the process image**
- **Initialize process control block**
  - many default values (ex: state is New, no I/O devices or files...)
- **Set up appropriate linkages**
  - Ex: add new process to linked list used for the scheduling queue

When to Switch a Process ?

- **A process switch may occur whenever the OS has gained control of CPU. ie when:**
  - Supervisor Call
    - explicit request by the program (ex: file open). The process will probably be blocked
  - Trap
    - An error resulted from the last instruction. It may cause the process to be moved to the Exit state
  - Interrupt
    - the cause is external to the execution of the current instruction. Control is transferred to IH

Mode Switching

- **It may happen that an interrupt does not produce a process switch**
- **The control can just return to the interrupted program**
- **Then only the processor state information needs to be saved on stack**
- **This is called mode switching (user to kernel mode when going into IH)**
- **Less overhead: no need to update the PCB like for process switching**

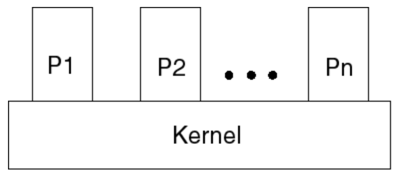
Steps in Process (Context) Switching

- Save context of processor including program counter and other registers
- Update the PCB of the running process with its new state and other associate info
- Move PCB to appropriate queue - ready, blocked
- Select another process for execution
- Update PCB of the selected process
- Restore CPU context from that of the selected process

Execution of the Operating System

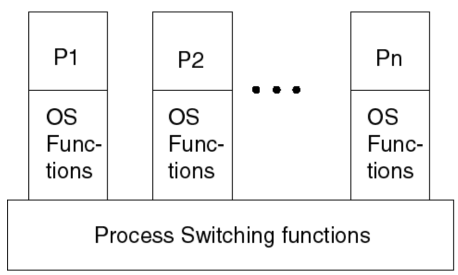
- Up to now, by process we were referring to “user process”
- If the OS is just like any other collection of programs, is the OS a process?
- If so, how it is controlled?
- The answer depends on the OS design.

Nonprocess Kernel (old)



- The concept of process applies only to user programs
- OS code is executed as a separate entity in privilege mode
- OS code never gets executed within a process

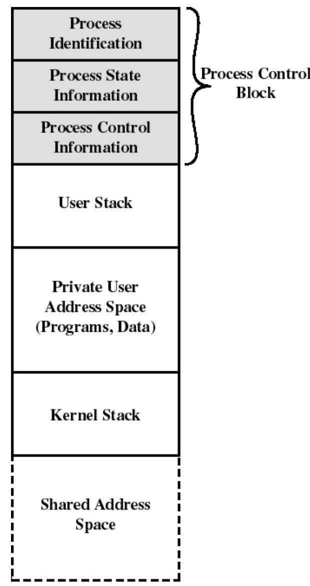
Execution within User Processes



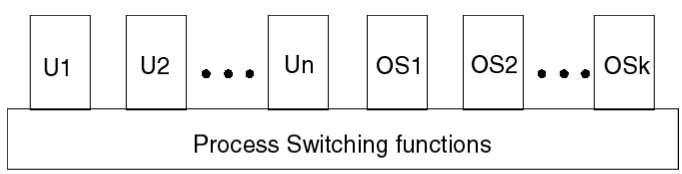
- Virtually all OS code gets executed within the context of a user process
- On Interrupts, Traps, System calls: the CPU switch to kernel mode to execute OS routine within the context of user process (mode switch)
- Control passes to process switching functions (outside processes) only when needed

Execution within User Processes

- OS code and data are in the shared address space and are shared by all user processes
- Separate kernel stack for calls/returns when the process is in kernel mode
- Within a user process, both user and OS programs may execute (more than 1)



Process-based Operating System



- The OS is a collection of system processes
- major kernel functions are separate processes
- small amount of process switching functions is executed outside of any process
- Design that easily makes use of multiprocessors