Topic 3

Process Description and Control

Topic Objectives

- Understand process states, state transition and their relationship with the dispatcher (process scheduler).
- Understand OS control structures, in particular, the process table and the role of PCB and its contents.
- Understand what constitutes the process image of a process.
- Understand mode switching and process switching and their difference.
- Be aware of the overall structure of UNIX SVR4's process management.
- Be aware of process related system calls in UNIX.

Readings

- Stallings: Chapter 3

Process

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by the execution of a sequence of instructions, a current state, and an associated set of system instructions

Process Elements

- Identifier
- State
- Priority
- Program counter
- Memory pointers
- Context data
- I/O status information
- Accounting information

Process Control Block

- Contains the process elements
- Created and managed by the operating system
- Allows support for multiple processes

Process Control Block

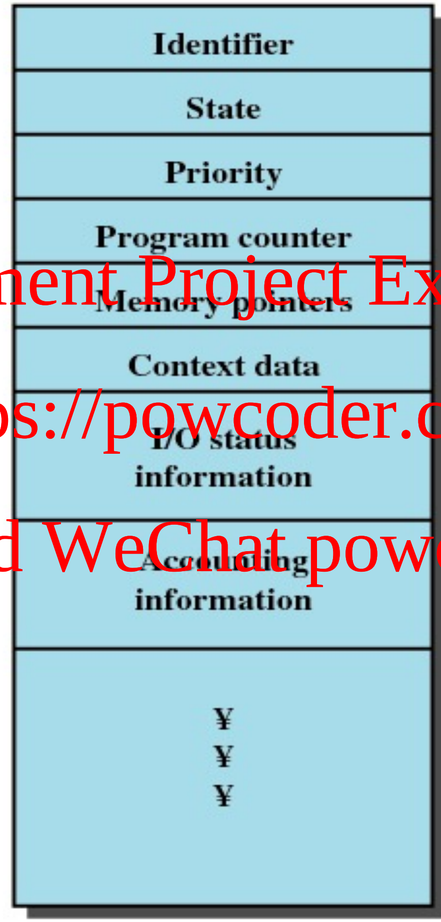


Figure 3.1 Simplified Process Control Block

Trace of Process

- Sequence of instructions that execute for a process
- Dispatcher switches the processor from one process to another

Example Execution

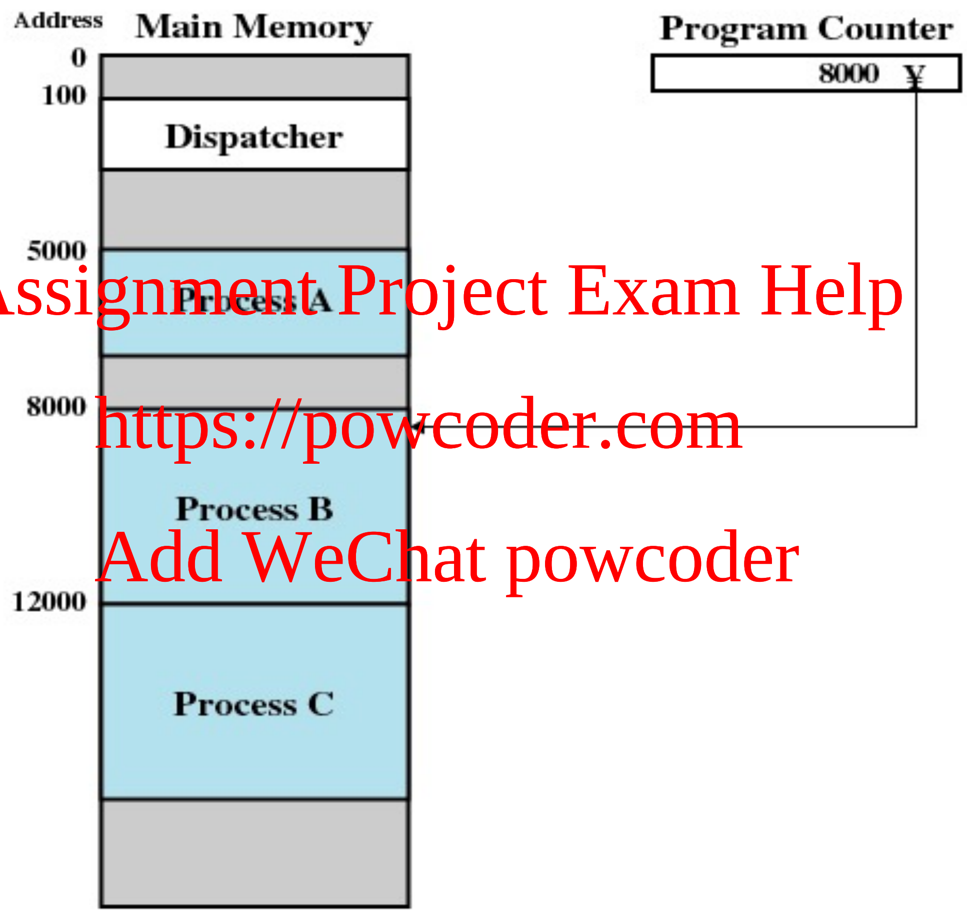


Figure 3.2 Snapshot of Example Execution (Figure 3.4) at Instruction Cycle 13

Trace of Processes

5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011
(a) Trace of Process A	**(b) Trace of Process B**	**(c) Trace of Process C**

5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Figure 3.3 Traces of Processes of Figure 3.2

1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
------------------ Time out	
7	100
8	101
9	102
10	103
11	104
12	105
13	8000
14	8001
15	8002
16	8003
--------------- I/O request	
17	100
18	101
19	102
20	103
21	104
22	105
23	12000
24	12001
25	12002
26	12003

27	12004
28	12005
------------------ Time out	
29	100
30	101
31	102
32	103
33	104
34	105
35	5006
36	5007
37	5008
38	5009
39	5010
40	5011
------------------ Time out	
41	100
42	101
43	102
44	103
45	104
46	105
47	12006
48	12007
49	12008
50	12009
51	12010
52	12011
------------------ Time out	

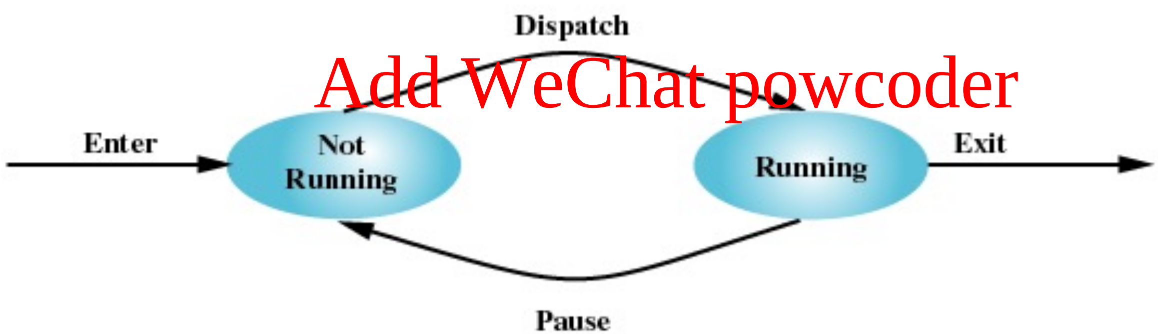
100 = Starting address of dispatcher program

shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

Figure 3.4 Combined Trace of Processes of Figure 3.2

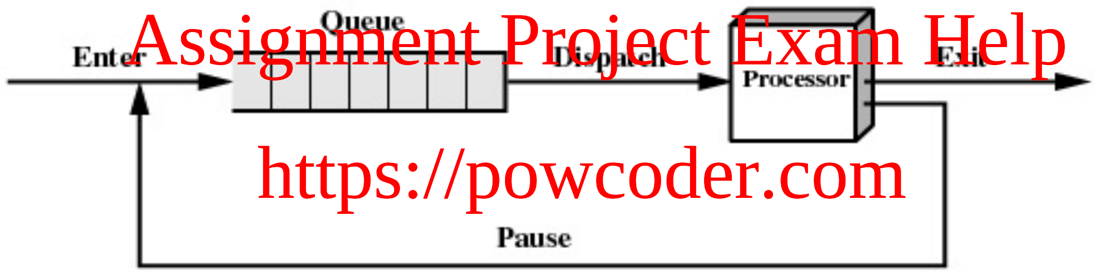
Two-State Process Model

- Process may be in one of two states
  - Running
  - Not-running



(a) State transition diagram

Not-Running Process in a Queue



(b) Queuing diagram

Process Creation

- New batch job
  - On old mainframe computers, the operating system is provided with a batch job control stream, usually on tape or disk. When the operating system is prepared to take on new work, it will read the next sequence of job control commands.

- Interactive logon
  - A user at a terminal logs on to the system

- Created by OS to provide a service
  - The operating system can create a process to perform a function on behalf of a user program, without the user having to wait (e.g., a process to control printing).

- Spawned by existing process
  - For purposes of modularity or to exploit parallelism, a user program can dictate the creation of a number of processes.

Process Termination

Table 3.2 Reasons for Process Termination

Normal completion	The process executes an OS service call to indicate that it has completed running.
Time limit exceeded	The process has run longer than the specified total time limit. There are a number of possibilities for the type of time that is measured. These include total elapsed time ("wall clock time"), amount of time spent executing, and, in the case of an interactive process, the amount of time since the user last provided any input.
Memory unavailable	The process requires more memory than the system can provide.
Bounds violation	The process tries to access a memory location that it is not allowed to access.
Protection error	The process attempts to use a resource such as a file that it is not allowed to use, or it tries to use it in an improper fashion, such as writing to a read-only file.
Arithmetic error	The process tries a prohibited computation, such as division by zero, or tries to store numbers larger than the hardware can accommodate.

Process Termination

Table 3.2 Reasons for Process Termination

Time overrun	The process has waited longer than a specified maximum for a certain event to occur.
I/O failure	An error occurs during input or output, such as inability to find a file, failure to read or write after a specified maximum number of tries (when, for example, a defective area is encountered on a tape), or invalid operation (such as reading from the line printer).
Invalid instruction	The process attempts to execute a nonexistent instruction (often a result of branching into a data area and attempting to execute the data).
Privileged instruction	The process attempts to use an instruction reserved for the operating system.
Data misuse	A piece of data is of the wrong type or is not initialized.
Operator or OS intervention	For some reason, the operator or the operating system has terminated the process (for example, if a deadlock exists).
Parent termination	When a parent terminates, the operating system may automatically terminate all of the offspring of that parent.
Parent request	A parent process typically has the authority to terminate any of its offspring.

Problem with Two-State Model

- Not-running
  - ready to execute
- Blocked
  - waiting for I/O
- Dispatcher cannot just select the process that has been in the queue the longest time because it may have been blocked

A Five-State Model

- Running
- Ready
- Blocked
- New
- Exit

Five-State Process Model

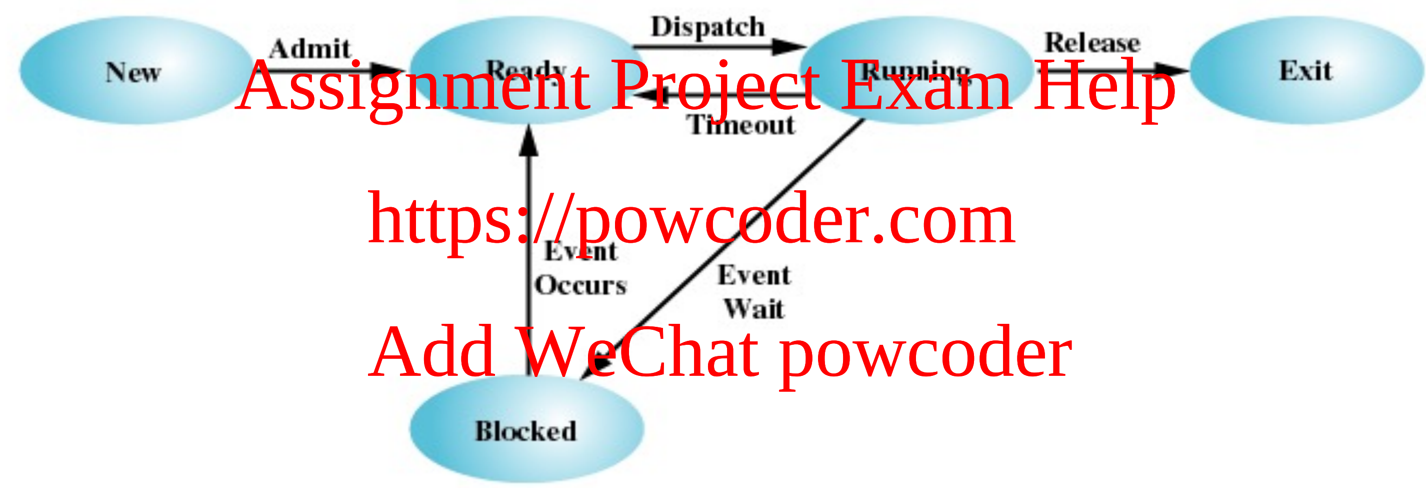
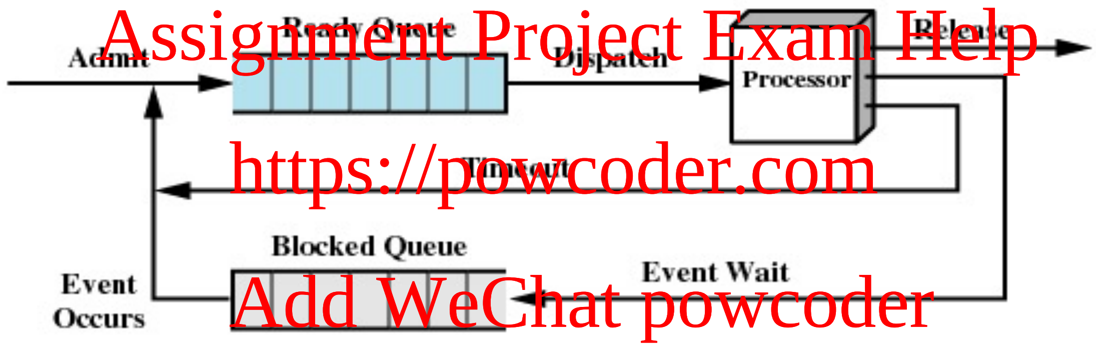


Figure 3.6 Five-State Process Model

Using Two Queues



(a) Single blocked queue

Process States

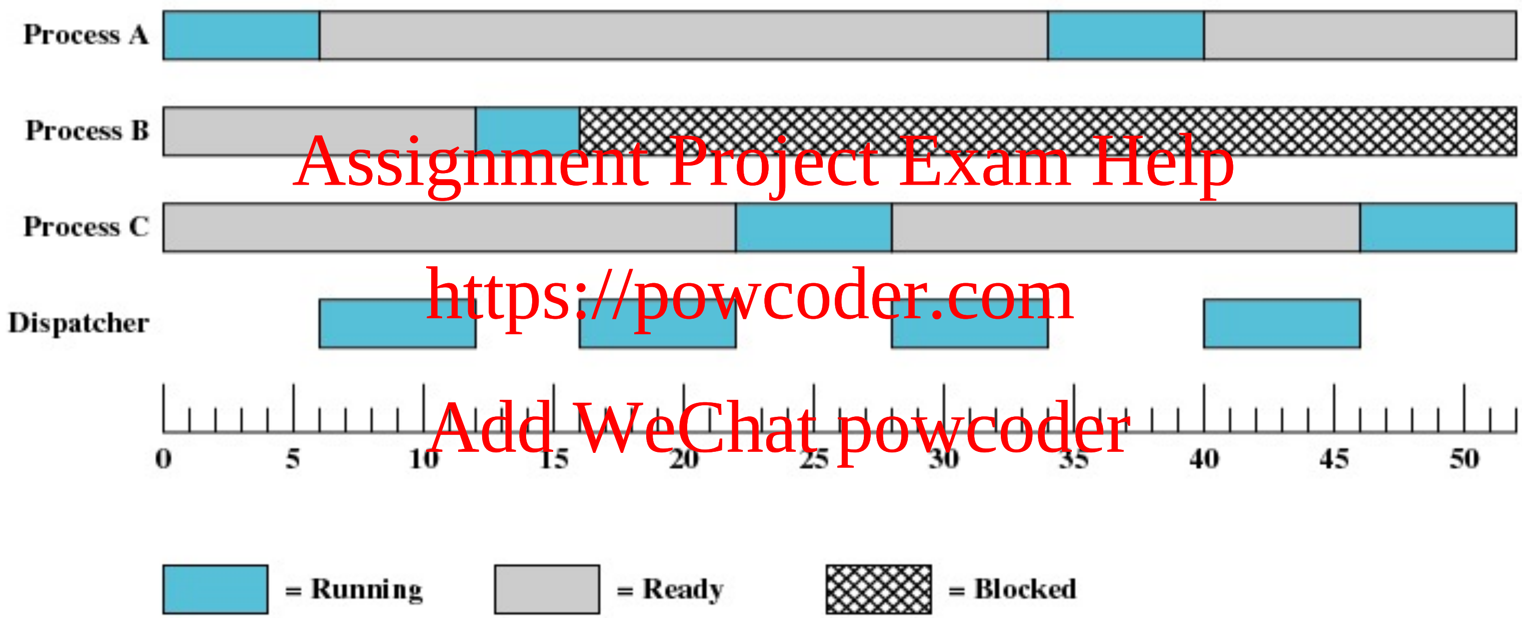
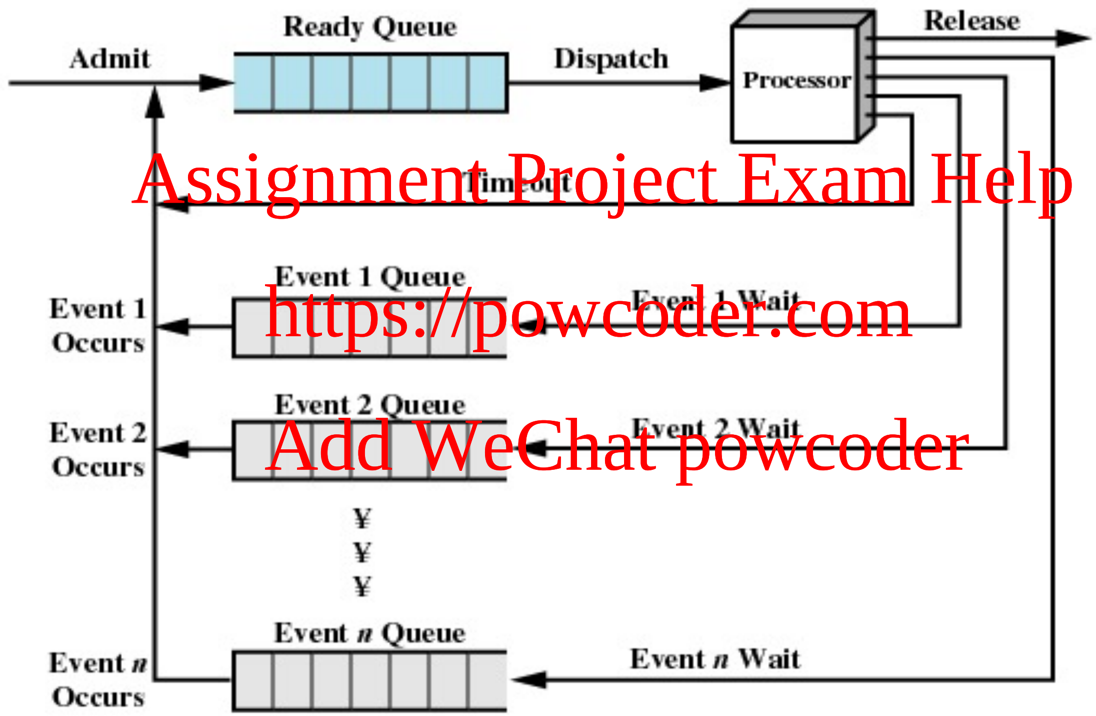


Figure 3.7 Process States for Trace of Figure 3.4

Multiple Blocked Queues



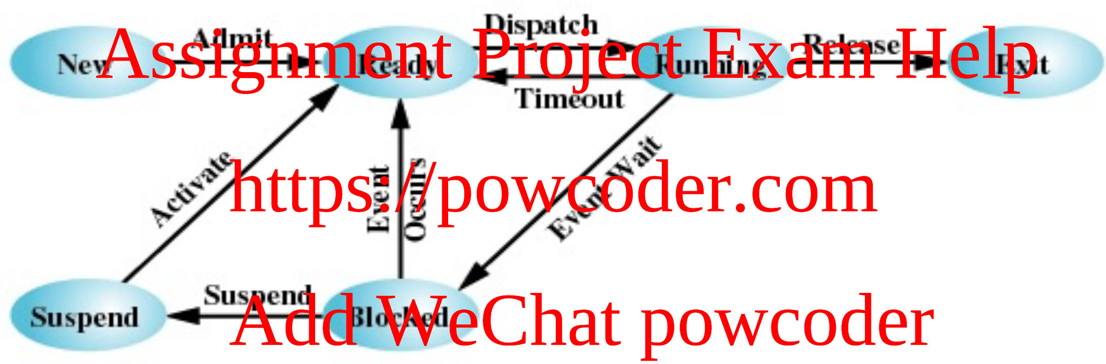
(b) Multiple blocked queues

Figure 3.8 Queuing Model for Figure 3.6

Suspended Processes

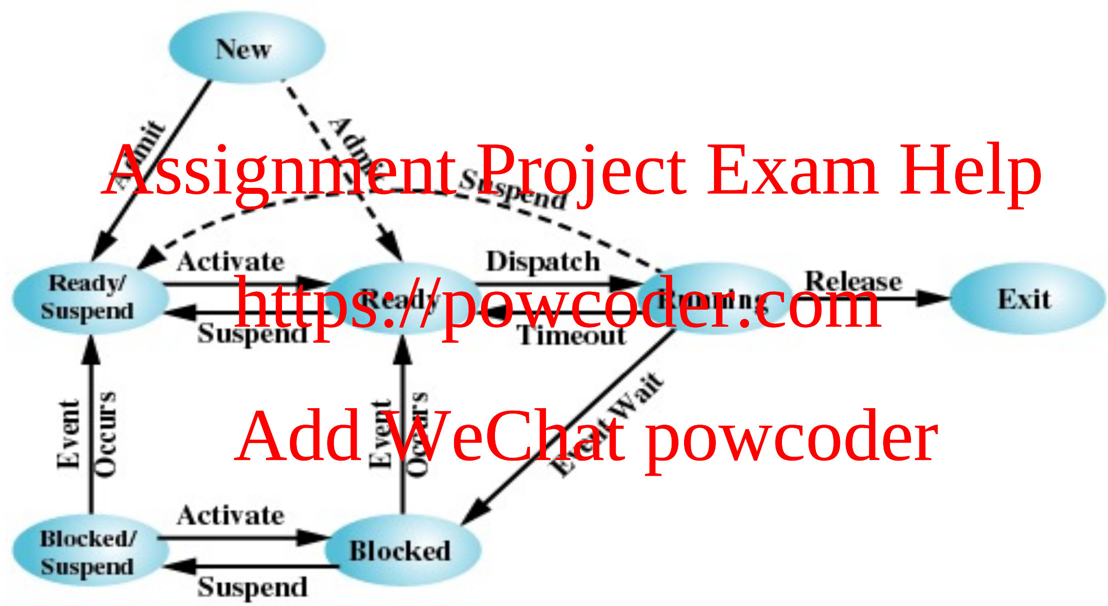
- Processor is faster than I/O so all processes could be waiting for I/O
- Swap these processes to disk to free up more memory
- Blocked state becomes suspend state when swapped to disk
- Two new states
  - Blocked/Suspend
  - Ready/Suspend

One Suspend State



(a) With One Suspend State

Two Suspend States



(b) With Two Suspend States

Figure 3.9 Process State Transition Diagram with Suspend States

Reasons for Process Suspension

Table 3.3 Reasons for Process Suspension

Swapping	The operating system needs to release sufficient main memory to bring in a process that is ready to execute.
Other OS reason	The operating system may suspend a background or utility process or a process that is suspected of causing a problem.
Interactive user request	A user may wish to suspend execution of a program for purposes of debugging or in connection with the use of a resource.
Timing	A process may be executed periodically (e.g., an accounting or system monitoring process) and may be suspended while waiting for the next time interval.
Parent process request	A parent process may wish to suspend execution of a descendent to examine or modify the suspended process, or to coordinate the activity of various descendents.

Operating System Control Structures

- Information about the current status of each process and resource
- Tables are constructed for each entity the operating system manages

Processes and Resources

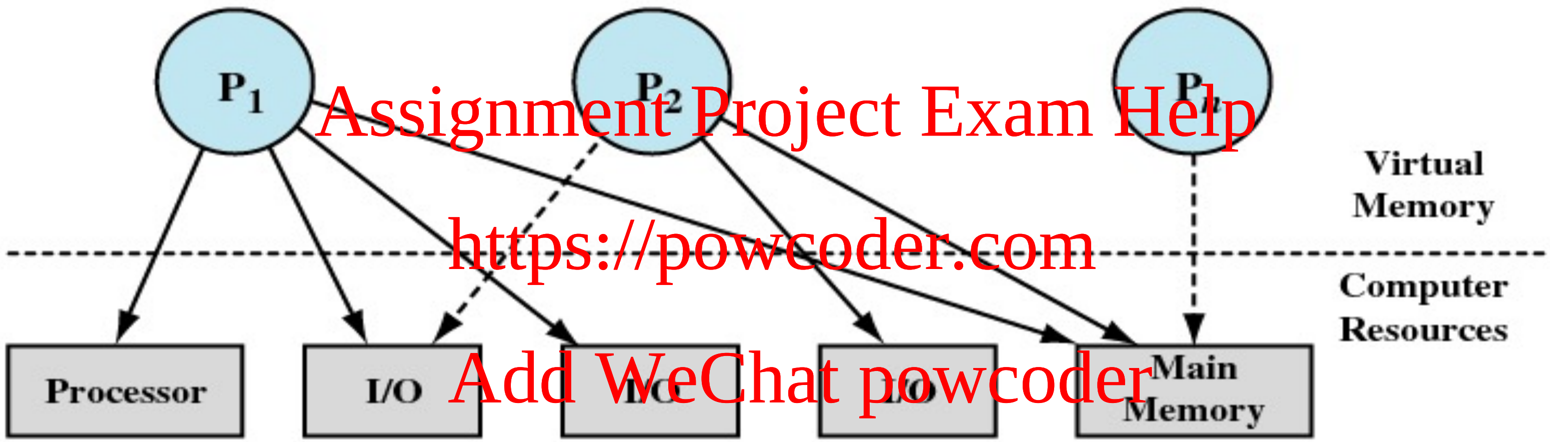


Figure 3.10 Processes and Resources (resource allocation at one snapshot in time)

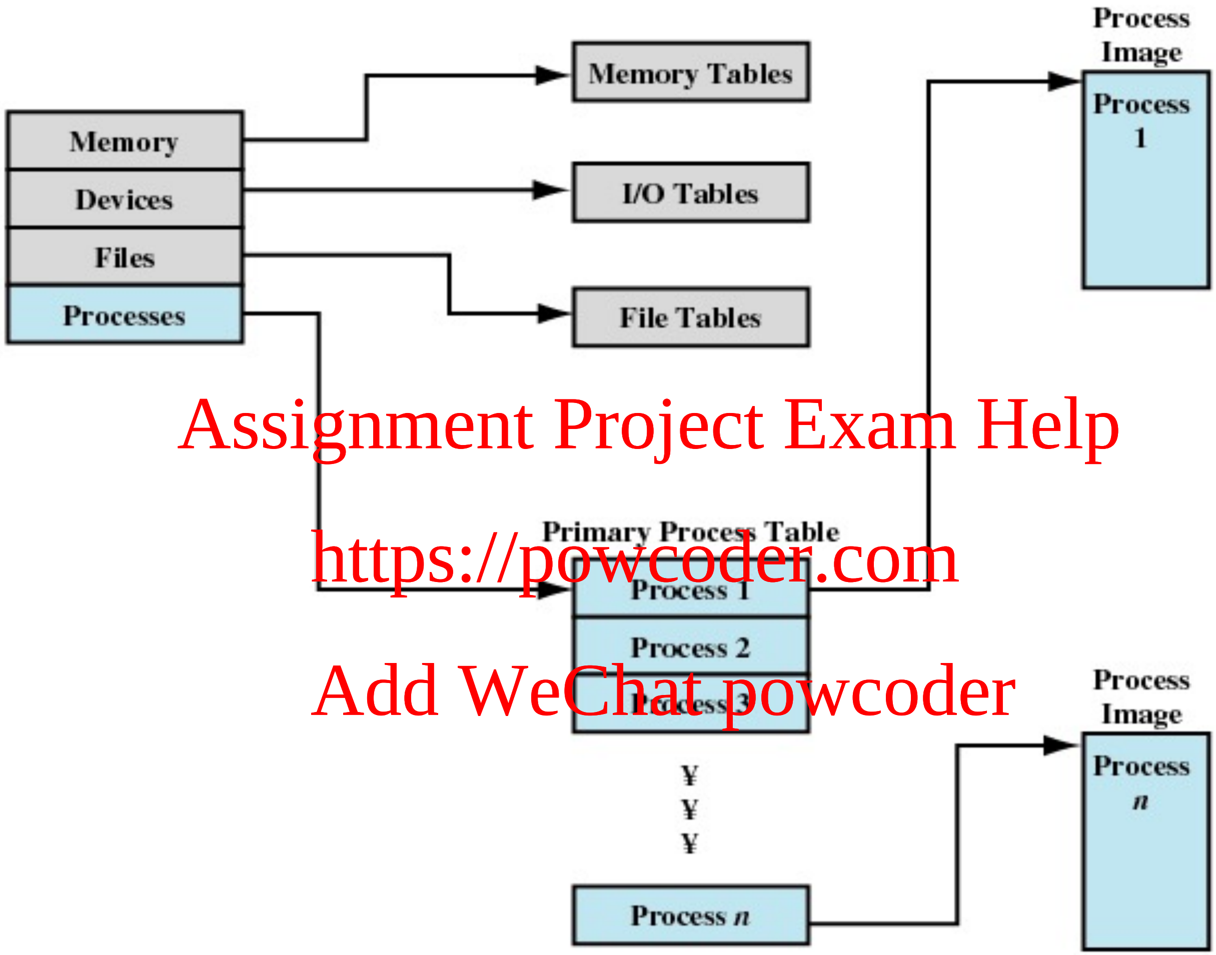


Figure 3.11 General Structure of Operating System Control Tables

Memory Tables

- Allocation of main memory to processes
- Allocation of secondary memory to processes
- Protection attributes for access to shared memory regions
- Information needed to manage virtual memory

I/O Tables

- I/O device is available or assigned
- Status of I/O operation
- Location in main memory being used as the source or destination of the I/O transfer

File Tables

- Existence of files
- Location on secondary memory
- Current Status
- Attributes
- Sometimes this information is maintained by a file management system

Process Table

- Where process is located
  - program
  - data
  - stack
- Attributes in the process control block
  - process identification
  - processor state
  - process control

Process Image

Table 3.4 Typical Elements of a Process Image

User Data

The modifiable part of the user space. May include program data, a user stack area, and programs that may be modified.

User Program

The program to be executed.

System Stack

Each process has one or more last-in-first-out (LIFO) system stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls.

Process Control Block

Data needed by the operating system to control the process (see Table 3.5).

Process Control Block

- Process identification
  - Identifiers
    - Numeric identifiers that may be stored with the process control block include
      - Identifier of this process
      - Identifier of the process that created this process (parent process)
      - User identifier

Processor State Information

- Processor State Information
  - Contents of processor registers
    - User-visible registers
    - Control and status registers (PC and PSW)
    - Stack pointers
  - Program status word (PSW)
    - Contains status information
    - Example: the EFLAGS register on Pentium machines

Pentium II EFLAGS Register

ID = Identification flag
VIP = Virtual interrupt pending
VIF = Virtual interrupt flag
AC = Alignment check
VM = Virtual 8086 mode
RF = Resume flag
NT = Nested task flag
IOPL = I/O privilege level
OF = Overflow flag

DF = Direction flag
IF = Interrupt enable flag
TF = Trap flag
SF = Sign flag
ZF = Zero flag
AF = Auxiliary carry flag
PF = Parity flag
CF = Carry flag

Figure 3.12 Pentium II EFLAGS Register

Process Control Block

- Process Control Information
  - Scheduling and State Information

    This is information that is needed by the operating system to perform its scheduling function. Typical items of information:

    - Process state: defines the readiness of the process to be scheduled for execution (e.g., running, ready, waiting, halted).

    - Priority: One or more fields may be used to describe the scheduling priority of the process. In some systems, several values are required (e.g., default, current, highest-allowable)

    - Scheduling-related information: This will depend on the scheduling algorithm used. Examples are the amount of time that the process has been waiting and the amount of time that the process executed the last time it was running.
    - Event: Identity of event the process is awaiting before it can be resumed
    - We will discuss processor scheduling in Topic 10.

Process Control Block

- Process Control Information
  - Interprocess Communication
    - Various flags, signals, and messages may be associated with communication between two independent processes. Some or all of this information may be maintained in the process control block.
    - We will learn to use various interprocess communication mechanisms in UNIX systems in Topic 7.
  - Process Privileges
    - Processes are granted privileges in terms of the memory that may be accessed and the types of instructions that may be executed. In addition, privileges may apply to the use of system utilities and services.

Process Control Block

- Process Control Information
  - Memory Management
    - This section may include pointers to segment and/or page tables that describe the virtual memory assigned to this process.
    - We will look at memory management issues in Topic 9.
  - Resource Ownership and Utilization
    - Resources controlled by the process may be indicated, such as opened files. A history of utilization of the processor or other resources may also be included; this information may be needed by the scheduler.

Process Control Block

- Process Control Information
  - Data Structuring
    - A process may be linked to other process in a queue, ring, or some other structure. For example, all processes in a waiting state for a particular priority level may be linked in a queue. A process may exhibit a parent-child (creator-created) relationship with another process. The process control block may contain pointers to other processes to support these structures.

Modes of Execution

- User mode
  - Less-privileged mode
  - User programs typically execute in this mode
- System mode, control mode, or kernel mode
  - More-privileged mode
  - Kernel of the operating system

Process Creation

- Assign a unique process identifier
- Allocate space for the process
- Initialize process control block
- Set up appropriate linkages
  - Ex: add new process to linked list used for scheduling queue
- Create or expand other data structures
  - Ex: maintain an accounting file

When to Switch a Process

- Clock interrupt
  - process has executed for the maximum allowable time slice

- I/O interrupt

- Memory fault

  - memory address is in virtual memory so it must be brought into main memory, eg page fault

When to Switch a Process

- Trap
  - error or exception occurred
  - may cause process to be moved to Exit state
- Supervisor call
  - such as file open

Switching Process

- Save context of processor including program counter and other registers
- Update the process control block of the process that is currently in the Running state
- Move process control block to appropriate queue – ready; blocked; ready/suspend
- Select another process for execution

Switching Process

- Update the process control block of the process selected

- Update memory-management data structures such as page table, a very expensive operation.

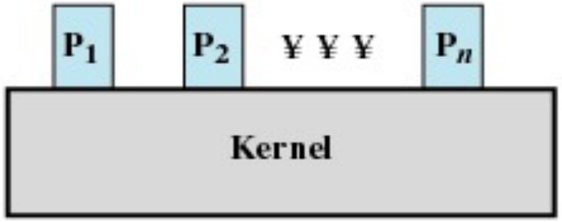
- Restore context of the selected process

Execution of the Operating System

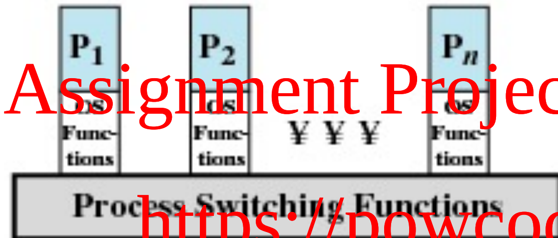
- Non-process Kernel
  - Execute kernel outside of any process
  - Operating system code is executed as a separate entity that operates in privileged mode
- Execution Within User Processes
  - Operating system software within context of a user process
  - Process executes in privileged mode when executing operating system code

Execution of the Operating System

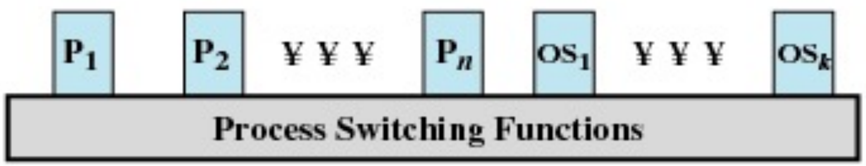
- Process-Based Operating System
  - Implement operating system as a collection of system processes
  - Useful in multi-processor or multi-computer environment



(a) Separate kernel



(b) OS functions execute within user processes



(c) OS functions execute as separate processes

Figure 3.15 Relationship Between Operating System and User Processes

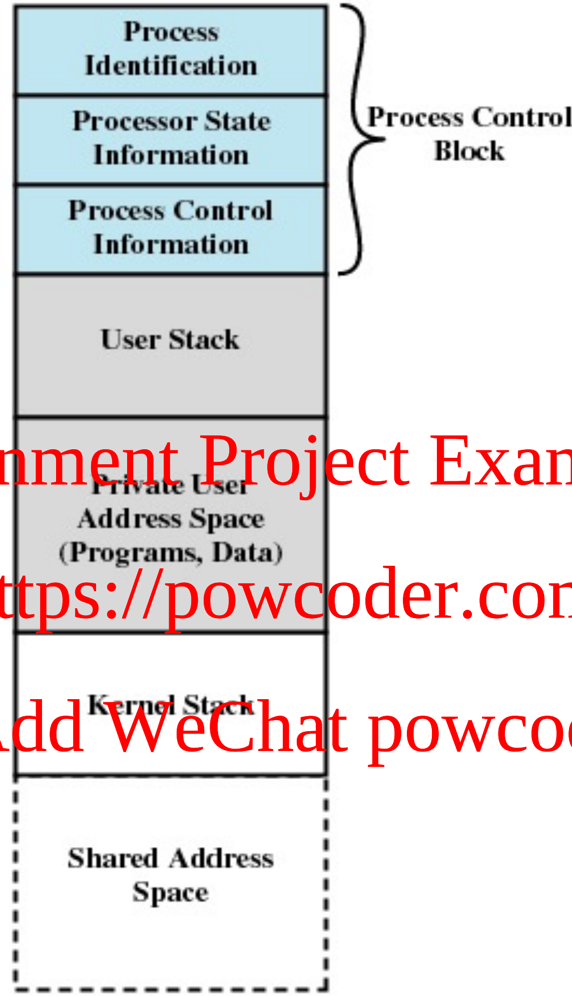
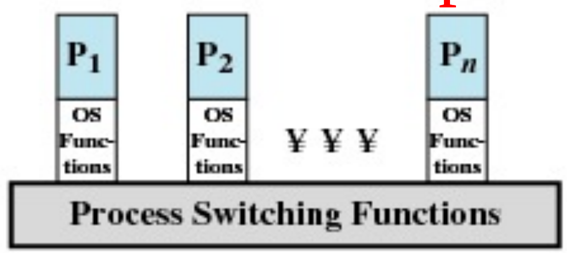


Figure 3.16 Process Image: Operating System Executes Within User Space

UNIX SVR4 Process Management

- Most of the operating system executes within the environment of a user process



(b) OS functions execute within user processes

UNIX Process States

Table 3.9 UNIX Process States

User Running	Executing in user mode.
Kernel Running	Executing in kernel mode.
Ready to Run, in Memory	Ready to run as soon as the kernel schedules it.
Asleep in Memory	Unable to execute until an event occurs; process is in main memory (a blocked state).
Ready to Run, Swapped	Process is ready to run, but the swapper must swap the process into main memory before the kernel can schedule it to execute.
Sleeping, Swapped	The process is awaiting an event and has been swapped to secondary storage (a blocked state).
Preempted	Process is returning from kernel to user mode, but the kernel preempts it and does a process switch to schedule another process.
Created	Process is newly created and not yet ready to run.
Zombie	Process no longer exists, but it leaves a record for its parent process to collect.

UNIX Process Image

Table 3.10 UNIX Process Image

User-Level Context	
Process Text	Executable machine instructions of the program
Process Data	Data accessible by the program of this process
User Stack	Contains the arguments, local variables, and pointers for functions executing in user mode
Shared Memory	Memory shared with other processes, used for interprocess communication
Register Context	
Program Counter	Address of next instruction to be executed; may be in kernel or user memory space of this process
Processor Status Register	Contains the hardware status at the time of preemption; contents and format are hardware dependent
Stack Pointer	Points to the top of the kernel or user stack, depending on the mode of operation at the time or preemption
General-Purpose Registers	Hardware dependent
System-Level Context	
Process Table Entry	Defines state of a process; this information is always accessible to the operating system
U (user) Area	Process control information that needs to be accessed only in the context of the process
Per Process Region Table	Defines the mapping from virtual to physical addresses; also contains a permission field that indicates the type of access allowed the process: read-only, read-write, or read-execute
Kernel Stack	Contains the stack frame of kernel procedures as the process executes in kernel mode

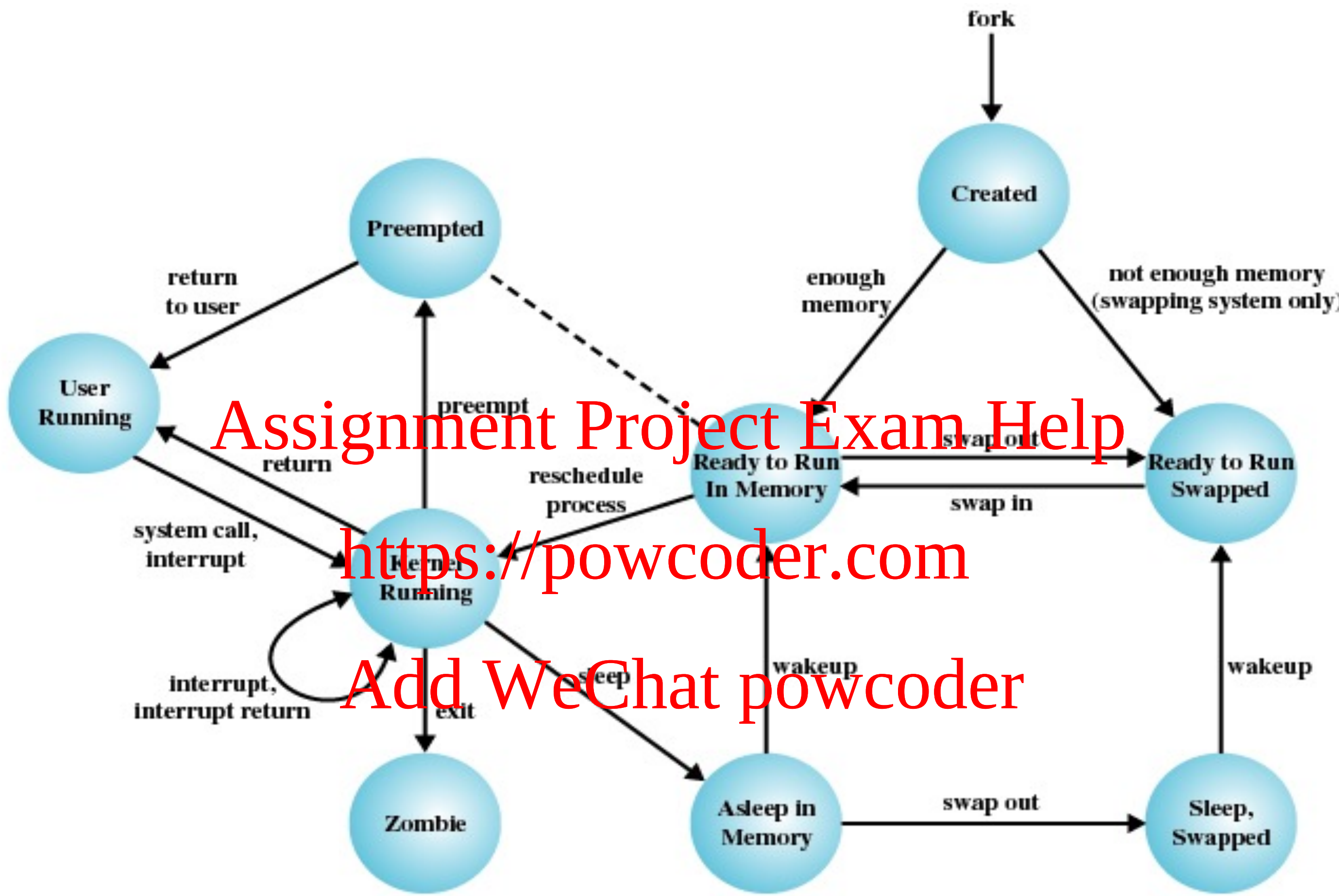


Figure 3.17 UNIX Process State Transition Diagram