- The process is the most important form of abstraction provided by modern OSes.
- Processes are used to ensure:
  - Resources are available to all programs.
  - The CPU is switched to multiple programs.
  - The CPU and I/O are used efficiently.
- A process is:
  - A program in execution it is actually the active program as only one program can use the CPU at a given time.
  - An instance of a program running on a computer
  - The entity that can be assigned to and executed on a processor.
  - An executed set of machine instructions.
  - A unit of activity characterised by a single sequential thread of execution.
- Programs are passive processes are active.
- The process uses two essential element of a program, the program code and a set of data associated to that code.

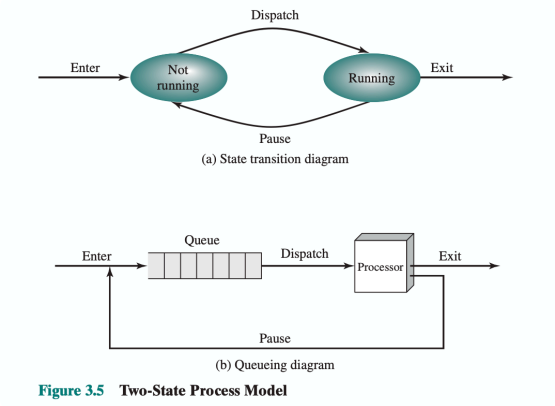


Figure 3.5 **Two-State Process Model**

- Interleaving processes
  - Trace:
    - The behaviour of an individual process can be seen as a sequence of instructions executed for that process.
    - The behaviour of the CPU can, itself be seen as interleaved traces of the various processes.
  - Dispatcher:
    - The switching of one process to another.
- Two state process-
  - Process is created
  - The process is always either running or not running
  - The process terminates.

- Reasons for the creation of processes
  - New batch job
    - In a batch environment the process is created as a response to the submission of a job.
  - Interactive logon
    - When a new user intends to logon a process is created.
  - The OS can create a process on behalf of an application, e.g to create a process to print on behalf of an application.

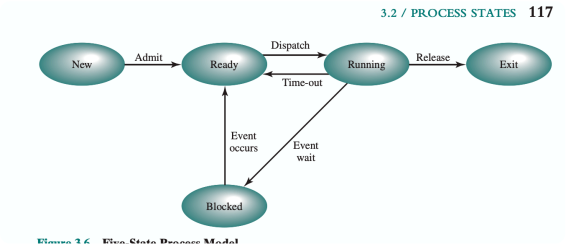
New batch job	The OS is provided with a batch job control stream, usually on tape or disk. When the OS is prepared to take on new work, it will read the next sequence of job control commands.
Interactive log-on	A user at a terminal logs on to the system.
Created by OS to provide a service	The OS can 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
- Computer systems need to indicate the completion of any program.
- The batch job does this with the halt instruction. The halt will cause an interrupt to indicate to the OS that the job has completed.

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.
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 (e.g., 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.

- If all processes are ready to be executed, then there will be a queue system as shown above. It will work in a round robin fashion where each process is allocated a time and then moved back to the queue, unless blocjed.
- At times the dispatcher will have to scan the processes and see which ones are block and allocate those that are not blocked and been there the longest in the queue.
- The not running state can be divided into two states; the blocked and not locked state.



- Running is the current process.
- New e.g. a new process which has been logged but not ready yet for execution.
- Blocked- waiting for an event to occur so that it can be executed.
- Exit- a process after termination.

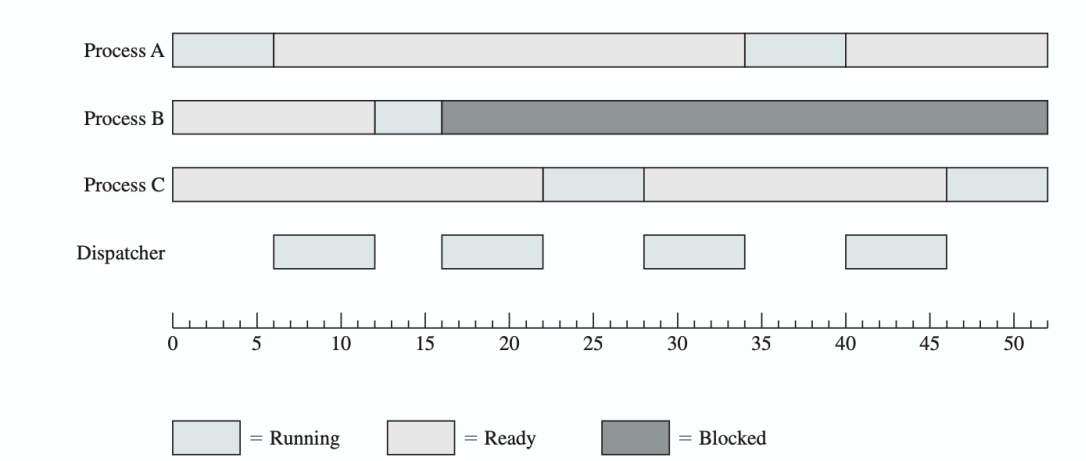


Figure 3.7 **Process States for the Trace of Figure 3.4**

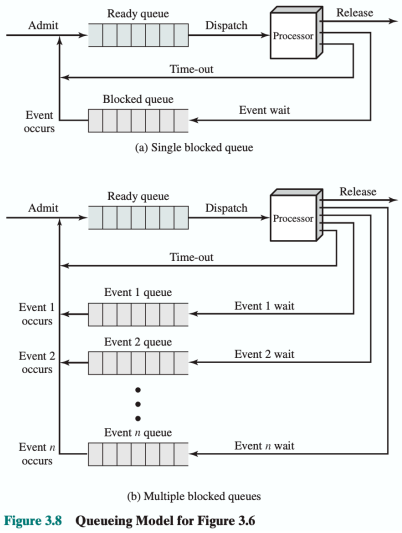


Figure 3.8 **Queueing Model for Figure 3.6**

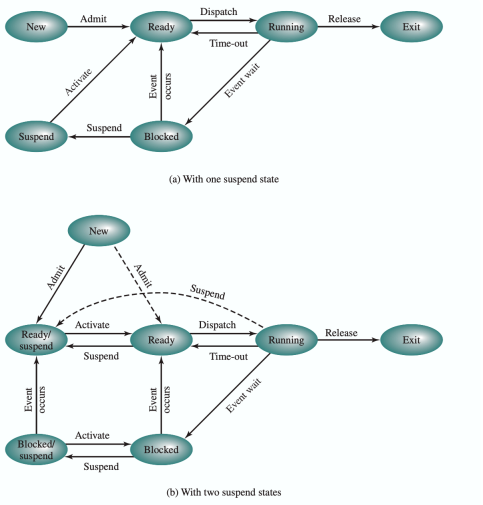


Figure 3.9 **Process State Transition Diagram with Suspend States**

- Figure 3.9 ; ready and blocked are in the main memory. Blocked/suspend (awaiting an event) and ready/suspend (just needs to be transferred) in the secondary memory.

Table 3.3 Reasons for Process Suspension

Swapping	The OS needs to release sufficient main memory to bring in a process that is ready to execute.
Other OS reason	The OS 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 descendants.

- A suspended process is not in the main memory, therefore cannot be executed until transferred.
- A swap may occur in order to shift a ready/suspended process into the main memory.
- The process may be placed in a suspended state by an agent or parent for debugging reasons.
- It can only be remove from this state when the agent explicitly orders the removal.

Another definition for OS
- That entity which manages the the use of system resources through processes.

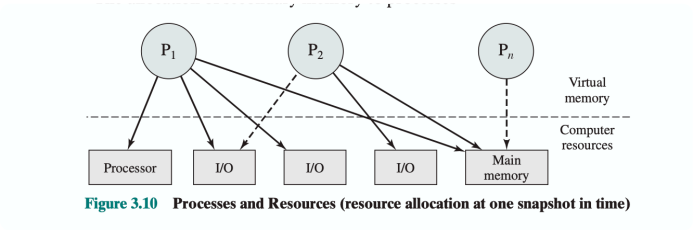


Figure 3.10 Processes and Resources (resource allocation at one snapshot in time)

- P1 is in the main memory with two I/O.
- P2 is blocked as it has one I/O which is shared with p1 and it is waiting for it.
- Pn is in the secondary memory therefore it is suspended.

OS structure

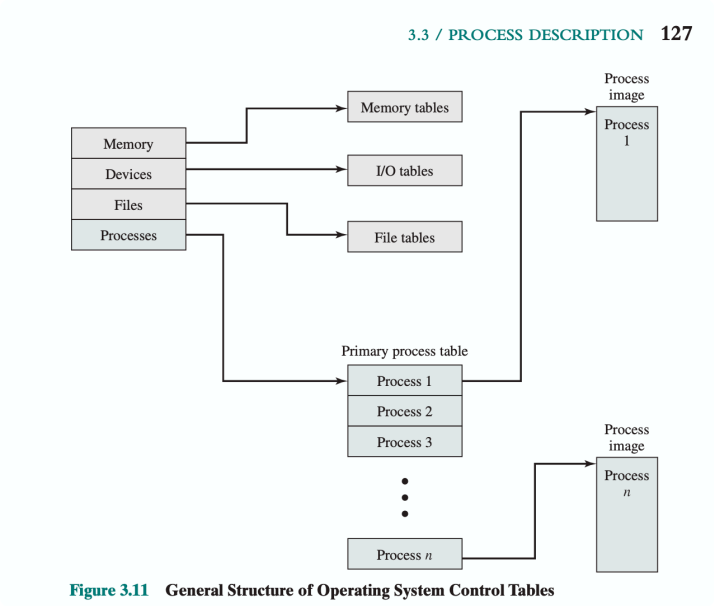


Figure 3.11 General Structure of Operating System Control Tables

- I/O tables are used to to manage I/O devices. An I/O device may be assigned to a given process at any time. The OS needs to know the status and location being used for the transfer between memory and I/O devices.
- File tables- information about the existence of files and their location on secondary memory, their current status etc. these can be maintained by a file management system.
- the I/O, memory and files are managed on behalf of processes. So there will be reference to these in the process tables.
- These tables need to be accessed by the OS, therefore, are subject to the memory management.

- The OS must know the location of processes and also the attributes of the process which are necessary for its management.
- Process location:
  - Processes hold a code from a program or programs.
  - There must be sufficient memory assigned to the process which can hold program data and the code.
  - A stack keeps track of prod=cedure callsand parameter passing between procedures. An execution of programs usually uses stacks.
- The OS uses attributes of the process for process control.
  - The process control block (PCB) is a compilation of the attributes.
  - The collection of program code, data, stack, and the PCB is the process image.
  - The OS has a memory management scheme which will determine the location of the 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.
Stack Each process has one or more last-in-first-out (LIFO) 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 OS to control the process (see Table 3.5).

- There are 3 main categories of the PCB.
  - Process identification.
  - Processor state information.
  - Process control information.

Table 3.5 Typical Elements of a 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
User-Visible Registers A user-visible register is one that may be referenced by means of the machine language that the processor executes while in user mode. Typically, there are from 8 to 32 of these registers, although some RISC implementations have over 100.
Control and Status Registers These are a variety of processor registers that are employed to control the operation of the processor. These include • **Program counter:** Contains the address of the next instruction to be fetched • **Condition codes:** Result of the most recent arithmetic or logical operation (e.g., sign, zero, carry, equal, overflow) • **Status information:** Includes interrupt enabled/disabled flags, execution mode
Stack Pointers 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. The stack pointer points to the top of the stack.

- Process identification:
  - Each process is assigned to a unique numeric identifier.
    - This may be a simple index into the primary process table.
    - Otherwise there must be a mapping which allows the OS to to locate the appropriate tables based on the process identifier.
- The identify may be used in several different ways.

    - Other tables may use the process identifier to cross reference process tables.
        - Memory tables may be organised to provide a map of main memory, and indicate the allocation of memory to processes.
        - Similar references will appear in I/O and file tables.
    - When processes interact with one another the process ID tells the OS about the location of the communication.
    - When a process creates another process the process ID will differentiate between parent and child processes.
- Processor state information
    - This consists of the contents of processor registers.
    - When a process is running all the information is in the registers, when interrupted it needs to be saved somewhere so that it can resume once interrupt is handles. The number of registers involved depends on the type of processor.
    - Typically the register will include 1) user-visible registers 2)control and status registers 3)stack pointers.
    - A set of registers are often known as Program Status Word (PSW). These contain status information.
        - They contain condition codes and other status information.
        - EFLAGS register is a good example which was run on x86 processor.
- Process control information
    - This controls and coordinates the active processes.
    - Information needed by the OS to perform its scheduling function.
        - Process state
        - Priority
        - Scheduling related information
        - iD of any event the process may be awaiting.
    - Links/pointers to other processes
    - Interprocess communication
    - Process privileges
    - Memory management
    - Resources controlled by the process.

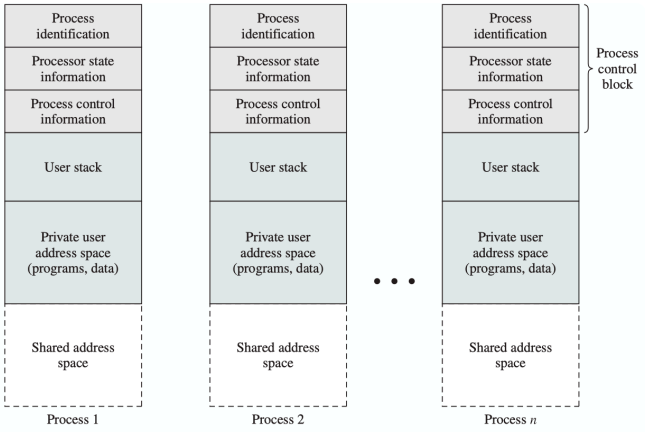


Figure 3.13 User Processes in Virtual Memory

- The PCB is the most important data structure in an OS. It contains all the information about a process which the OS needs.
- Every module of the OS uses these blocks to read/modify. This leads to an important design issue.
- The difficulty lies in the protection of these tables. Access ID not an issue.
    - A bug in a single routine, such as an interrupt handler, could damage PCB, which in turn can affect the systems ability to manage the affected processes.
    - A design change of the PCB can affect the modules of the OS.

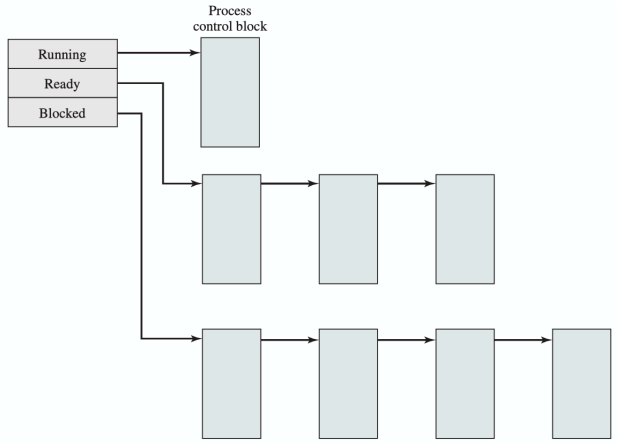


Figure 3.14 Process List Structures

OS control

- Most processors support at least two modes of execution. Privileged mode and user mode.
    - Privileged mode includes reading and altering registers. This is referred to as the system mode, kernel mode or control mode.
    - User mode normally execute user programs.
- The use of two modes is to stop interference of the PCB from user programs.
    - The kernel mode, the software has full control of the processor.

Process creation
- Assign a unique process ID to the new process.
- Allocate space
- Initialise the pcb.
- Set the appropriate linkages.
- Create or expand other data structures.

Interrupts, Traps and Supervisor calls

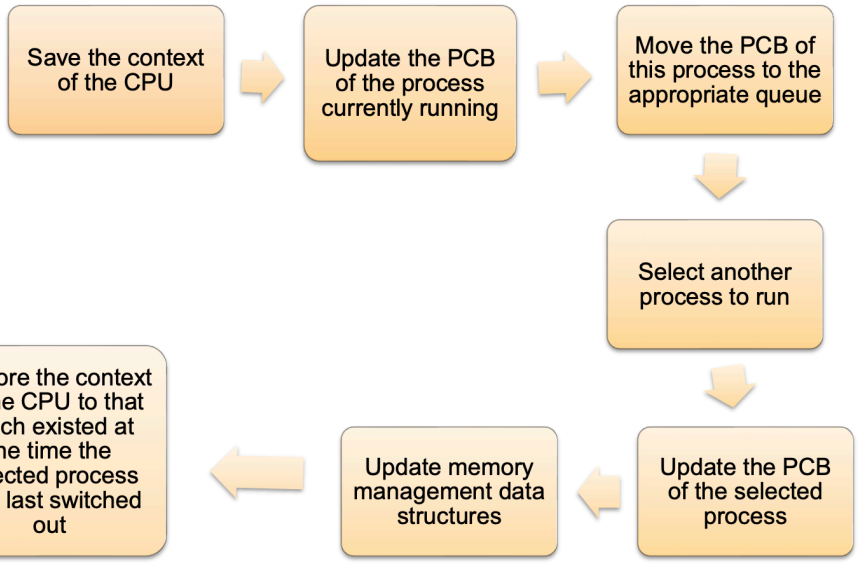
Table 3.8 Mechanisms for Interrupting the Execution of a Process

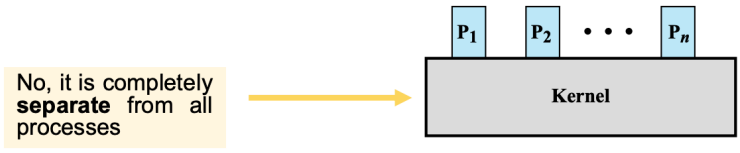
Mechanism	Cause	Use
Interrupt	External to the execution of the current instruction	Reaction to an asynchronous external event
Trap	Associated with the execution of the current instruction	Handling of an error or an exception condition
Supervisor call	Explicit request	Call to an operating system function

- Interrupts
  - Due to an event which is external to the current process
    - Timer interrupt
    - I/O interrupt
    - Memory fault
- Trap
  - A software interrupt
  - Error in the currently running process
  - OS will determine if the condition is fatal-
    - If yes, exit and next
    - If no, nature of error will determine next stage.
- Supervisor call
  - A call to an OS function. Made by a programm which requires a privileged task to be performed.
- Interrupts are asynchronus, while traps and SC are synchronous.

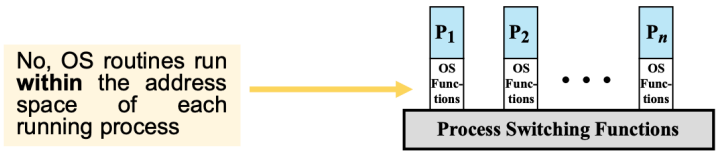
Mode Switching
- If an interrupt is pending, the processor
  - Sets the program counter to the stating address of an interrupt handler program.
  - Switches from user mode to kernel mode so that the interrupt processing code may have the privileged instructions.
- If no interrupts:
  - Proceeds to the fetch stat=ge and fetches the next instruction of the current program in the current process.
-

If the currently running process is to be swapped with another, then the OS does the following:

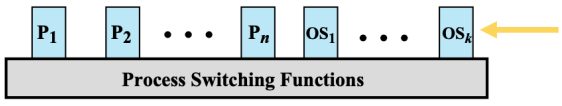




(a) Separate kernel



(b) OS functions execute within user processes



(c) OS functions execute as separate processes

Figure 3.15 Relationship Between Operating System and User Processes

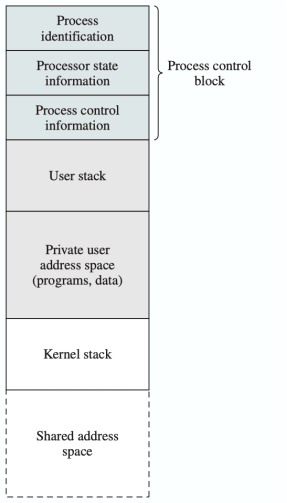


Figure 3.16 **Process Image: Operating System Executes within User Space**

- Unix (system v)
- Uses the model where most of the OS executes within the user mode.
- Certain system processes run in kernel mode