# Basics of Virtual Machine And Process

**This is an article on *Basics of Virtual Machine And Process* in *Operating System***.

Operating system is large, but modularity is important thus a computer system is made up of layers. The hardware is the lowest level. The kernel running at the next level uses the hardware instructions to create a set of system calls. Virtual machine creates an efficient, dynamic, secure and portable object-oriented facility. Virtual machine provides an interface. The resources of the physical computer are shared to create the virtual machines.

For example if the physical machine has three disk drives to support seven virtual machines, it can not allocate a disk drive to each virtual machine, but it will provide virtual disks. Virtual machine concept is difficult to implement because underlying machine has two modes user-mode and monitor-mode. Virtual machine software can run in monitor mode and execute in user mode.

**Benefits**

- Complete protection of the resources i.e., there is no direct sharing of resources, it is implemented by software.
- Virtual machine is completely isolated from all other virtual machines, so there are no security problems.
- Virtual machine system is a perfect vehicle for operating systems research and development.
  When to change operating system, current system must be stopped and taken out of use while changes are made and tested. This period is called "system development time".

## SYSTEM DESIGN AND IMPLEMENTATION

**Design Goals**

Design of the system will affect the choice of hardware and type of system i.e., batch, time-shared, single-user, multi-user, distributed, real-time or general purpose.

Requirements can be divided into two basic groups.

- **User Goals**
  User desire is system should be easy to learn, easy to use, reliable, safe and fast.
- **System goals**
  Similar set of requirements are there for the people who must design, create, maintain and

operate the system, i.e., the operating system should be easy to design, implement, and maintain. It should be flexible, reliable, error free and efficient.

**Mechanisms and Policies**

Mechanisms determine how to do something. For example mechanism for ensuring CPU protection is the timer construct. That is the decision of how long the timer is set for a particular user is a policy decision.

Policies decide what will be done. Policies are likely to change from place to place or time to time. Each change in policy require a change in the underlying mechanism. Policy decisions are important for all resource allocation and scheduling problems.

**Implementation**

Traditionally operating systems have been written in assembly language. But now operating systems can even be written in higher-level languages.

**Examples**

- First higher-level languages operating system is "MCP" (Master Control Program) for Burroughs Computers.
- "MULTICS" developed at MIT written in ‘PL/1’.
- "Prime OS" for prime computers written in "FORTRAN".
- "UNIX", "OS/2" and "Windows/NT" were written in ‘C’.

**Advantages of Higher-level languages**

- Code can be written faster, more compact, easier to understand and debug.
- Operating system is easier to port i.e., to move to some other hardware.

**Disadvantages of Higher-level languages**

- Reduced speed and increased storage requirements.
- Memory manager and the CPU Scheduler are critical.
- Bottleneck routines can be identified and replaced with assembly-language equivalents.

# PROCESS CONCEPTS

Process is nothing but a program in execution, or the instance of a program (or job) in execution. The terms job and process are used interchangebly. The term process has superseded. Major requirements of the operating system are to maximize processor use, minimize response time, avoiding deadlocks, and support inter process communication.

# PROCESS STATES

The principal function of a processor is to execute machine instructions residing in main memory. The execution of an individual program is referred to as a process or task. The life of a process is bounded by its creation and termination.

### Creation of a Process

When a new process is to be added, operating system immediately allocates the address space.

### Reasons for Process Creation

- Submission of a new batch job.
- Interactive log on by a user at a terminal.
- Created by operating system to provide a service.
- One process to cause the creation of another process.

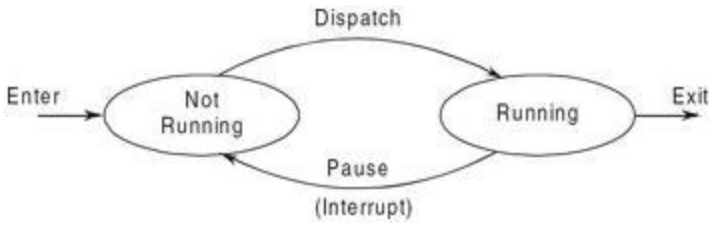
### Termination of Process

### Reasons for termination of a process

- Normal completion
- Time limit exceded
- Memory unavailable
- Bounds violation
- Protection error
- Arithmetic error
- Time overrun
- I/O failure
- Parent termination
- Halt instruction.

### Two-state Process Model

The main responsibility of the operating system is to control the execution of processes. The figure below shows at any time a process is either running or not-running state. When operating system creates a new process, first it enters in the not-running state and is waiting for a chance to execute.

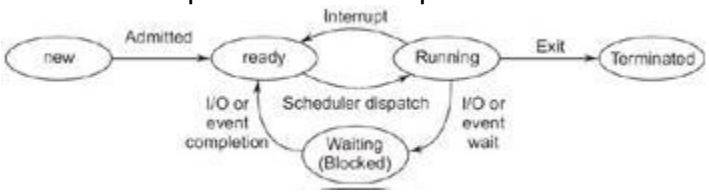



From time to time currently running process will be interrupted and the operating system will select a new process for running. The earlier process moves from the Running State to the not-running state. All these processes which are not running are kept in some sort of queue, waiting their chance to execute. This is Shown in the figure above..

**Five-state Process Model**

In round-robin fashion, on the available processes each process in the queue is given a certain amount of time to execute and then returned to the queue unless waiting (i.e., blocked).
The figure below shows some processes in the not-running state are ready to execute, where as others are blocked, i.e., waiting for an Input/Output operation to complete. Dispatcher is to scan for process that is not blocked (waiting) for dispatch. To handle this situation split the not-running state into two states ready and waiting (blocked).

- Running –> process is in execution
- Ready –> process prepared (ready) to execute
- Blocked (waiting) –> process cannot execute until some event occurs, like completion of an I/O operation.
- New –> process that has just created.
- Terminate –> process has completed its execution.




**Process State Transitions**

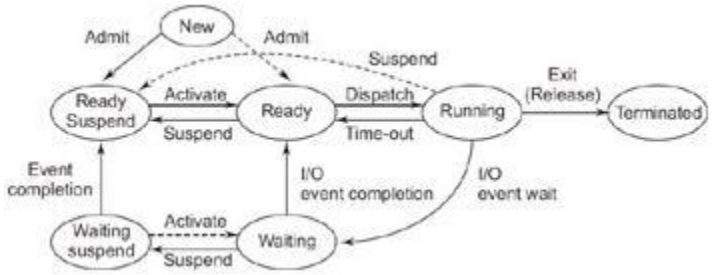
- Null-->New
- Ready-->Running
- Running-->Ready
- Waiting-->Ready
- New-->Ready
- Running-->terminated
- Running-->waiting

**Suspended Processes**

For processing more processes quickly, accommodate more process in main memory, for this expand the main memory. There are two problems in expansion of main memory. These are as follows:

- There is a cost associated with main memory
- Larger memory results in larger processes, not more processes.

Solution for the above problems is swapping, that is move part or all of a process from main memory to disk by suspend. When no process in main memory is in the ready state, then operating system swaps back to main memory. Swapping will enhance the performance. This situation is shown in the figure below:




In the above Figure dashed line indicates possible transitions but it is not necessary. When the operating system has performed a swapping out operation, it has two choices for selecting a process to bring into main memory. That is it can admit a newly created process or it can admit a previously suspended process. But preference is for a previously suspended process instead of new process to reduce the load on the system.

Ready : Process is in main memory and also available for execution.Waiting : Process is in main memory and awaiting for an event.Waiting, Suspend : Process is in secondary memory and awaiting for an event.Ready, Suspend : Process is in secondary memory and is available for execution when it is to be loaded into main memory.

**State Transitions**

- Waiting --> waiting, suspend
- Waiting, suspend --> Ready, suspend
- Ready, suspend --> Ready
- Ready --> Ready, suspend
- New --> Ready, suspend
- New --> Ready
- Waiting, suspend --> Waiting
- Runninig --> Ready, suspend

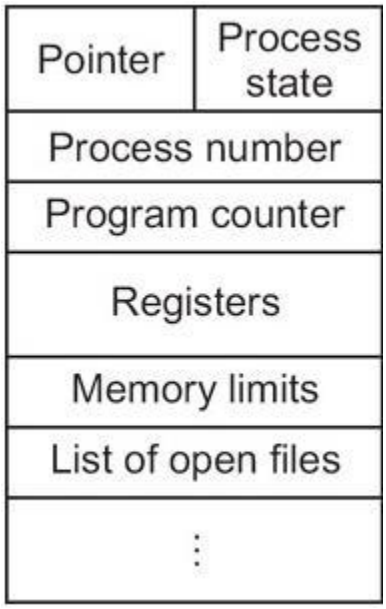
- Running --> Ready
- Ready --> Running
- Running --> Terminate

**Reasons for Process Suspension**

- For release of sufficient main memory
- Process causing a problem
- User suspend execution of a program
- Executed periodically
- Parent process may suspend execution of a child process

## PROCESS CONTROL BLOCK (PCB)

The collection of attributes related to a process hold in a block called **process control block or task control block or process descriptor or task descriptor**. This collection of program, data, stack and attributes is the process image. This is shown in the figure below:




**Elements of a Process Image**

- User data
- User program
- System stack
- Process control block

The process image is maintained as a contiguous block of memory. This block is

maintained in secondary memory. For execution of a process, the entire process image should be loaded into main memory. In modern operating systems, these image blocks need not be stored contiguously. Basing on the technique used, these blocks may be of variable length (called segments) or fixed length (called pages) or a combination.

**Process Attributes**

Different systems will organize the process control block information in different ways:

- **Process number or identification**

  Each process is assigned a unique numeric identifier. The identifier may be an index into the primary process table. Similar references will appear in Input/Output and file tables.
  Identifiers:
  - Identifier of this process
  - Identifier of the parent process
  - User identifier
- **Processor state information**

  It contains the contents of processor registers.
  - User Visible Registers
    These registers are referenced by means of the machine language.
  - Control and Status Registers
    Program counter,Condition codes (zero, sign, carry),Status information (interrupt enable, disable)
  - Stack pointers
    Stack pointer points to the top of the stack, in Last in First Out (LIFO) system. Stack store parameters and calling addresses.

- **Process Control Information**

  It contains the additional information needed for the operating system to control and co-ordinate various active processes
  - Scheduling and state information
  - Data structuring
  - Interprocess communication
  - Process privileges
  - Memory management
  - Resource ownership and utilization
    Even though the process control block is most important and central data structure in operating system it suffers with two problems.

- A bug in a single routine (like interrupt handler) can damage process control block, which can destroy the systems ability to manage the affected processes.
- A design change in the structure or semantics of the PCB can affect the number of modules in the operating system.

  In the next article I will discuss about process scheduling process carried out by CPU,cooperating processes and threads.

**Source: http://www.go4expert.com/articles/basics-virtual-machine-process-t22280/**