Process Management

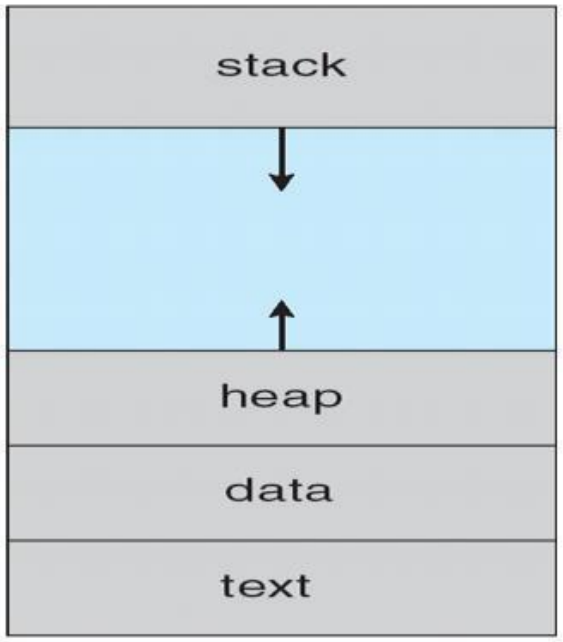
Process Concept

- An operating system executes a variety of programs
  - batch systems - jobs
  - time-shared systems - user programs or tasks
  - Job, task and program used interchangeably
- Process - a program in execution
  - process execution proceeds in a sequential fashion
- A process contains
  - program counter, stack and data section

Process vs Program

- program is a group of instructions to carry out a specified task.
- A program is a **passive entity,** for example, a file accommodating a group of instructions to be executed (executable file).
- Program is stored on disk in some file and does not require any other resources.
- When a program is executed, it is known as process.
- It is considered as an **active entity** and realizes the actions specified in a program.
- Process holds resources such as CPU, memory address, disk, I/O etc.
- Multiple processes can be related to the same program. It handles the operating system activities through **PCB (Process control Block)** which includes program counter, stack, state etc
- A web browser launches multiple processes, e.g., one per tab

- **Text section** is made up of the compiled program code, read in from non-volatile storage when the program is launched.
- The **Data section** is made up the global and static variables, allocated and initialized prior to executing the main.
- The **Heap** is used for the dynamic memory allocation, and is managed via calls to new, delete, malloc, free, etc.
- The **Stack** is used for local variables. Space on the stack is reserved for local variables when they are declared.



Process in Memory

The Process Model

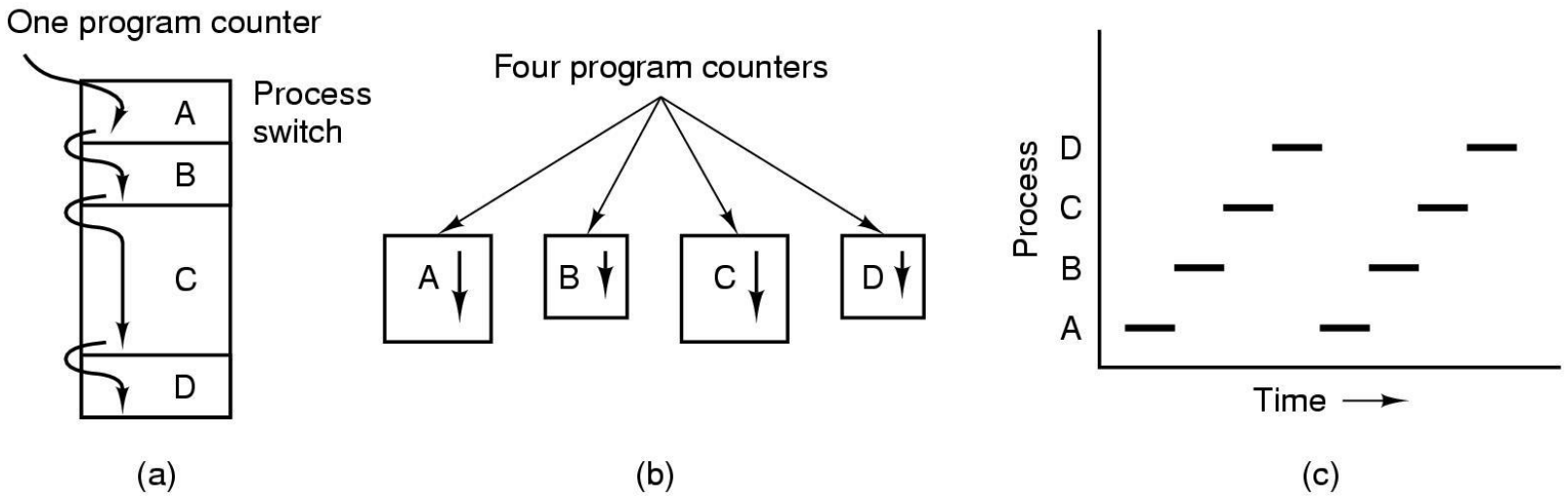


Figure 3-1. (a) Multiprogramming of four programs. (b) Conceptual model of four independent, sequential processes. (c) Only one program is active at once.

- Conceptually, every process has its own virtual CPU but in reality CPU switches back and forth from process to process.
- This Rapid switching back and forth is called **"Multi-Programming.**"

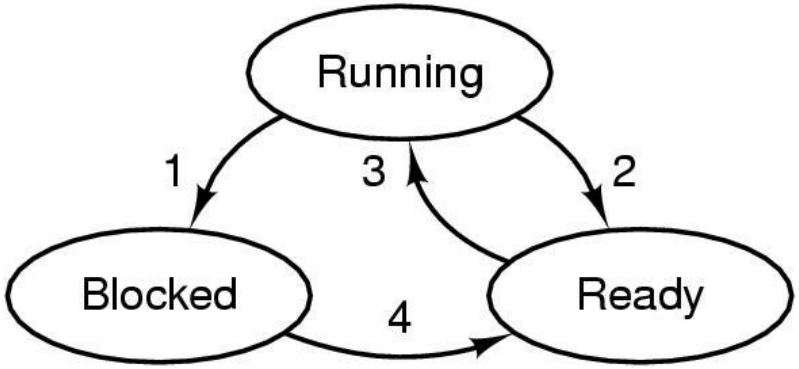
Multi-Programming

- In this there is 4 program in memory. We have single shared processor by all programs.
- There is only one program counter all programs in memory. In this program counter is initialized to execute the part of program A then with the help of process switch it transfer control to program B and execute some part of this.
- After that program counter for program C is active and execute some part of it and so onto program D. Then all the steps continuous may be randomly with the help of process switches until all program is not completed.

One Program At One Time

- At any given instant only one process run and other process after some interval of time.
- In other point of view all processes progress but only process actually runs at given instant of time.

Process States



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

Three-state process model is constituted of **READY**, **RUNNING** & **WAITING**.

Running: the process is currently executed by the CPU

Blocked: the process is waiting for a resource to come available

Ready: the process is ready to be selected.

Five state process models

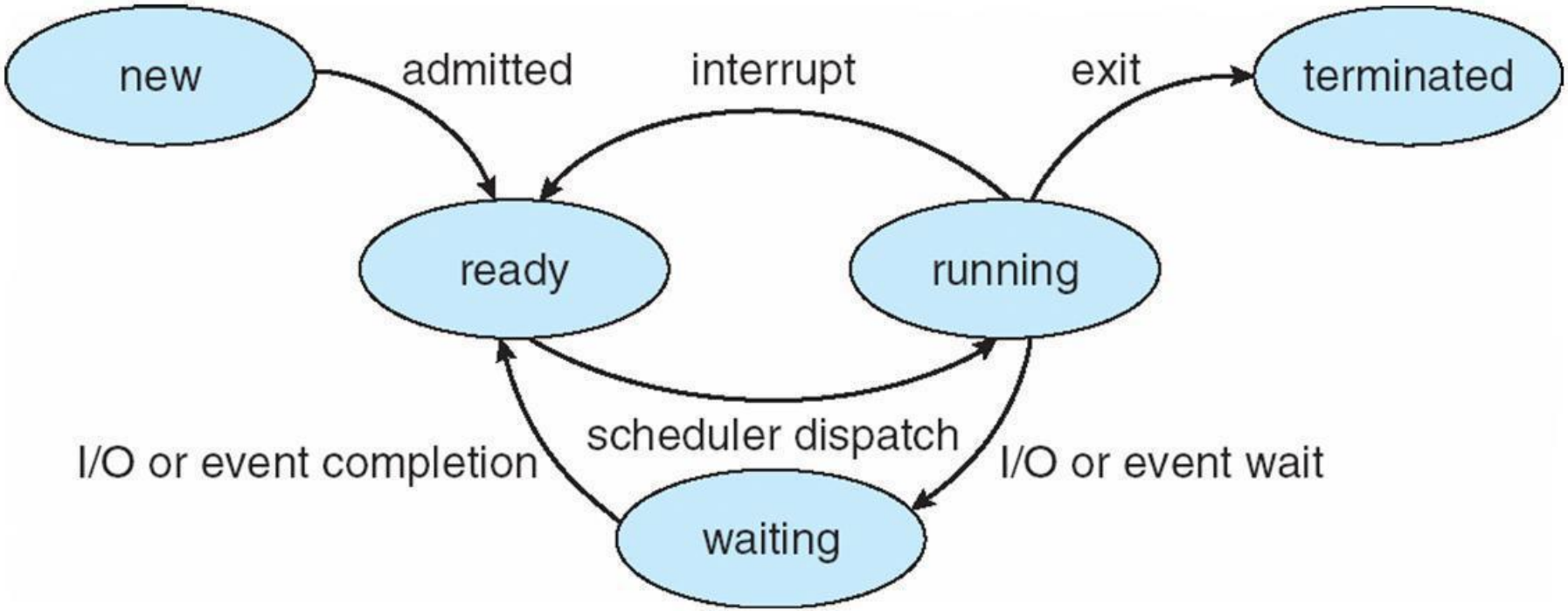


Figure 3-2. five-state Process State Transition Diagram

New : The process is being created.

Running : Instructions are being executed.

Waiting : Waiting for some event to occur(such as an I/O completion or reception of a signal).

Ready : Waiting to be assigned to a processor.

Terminated : Process has finished execution.

- **Preemptive** scheduling allows a running **process** to be interrupted by a high priority **process**.
- **Non-preemptive** scheduling, any new **process** has to wait until the running **process** finishes its CPU cycle.

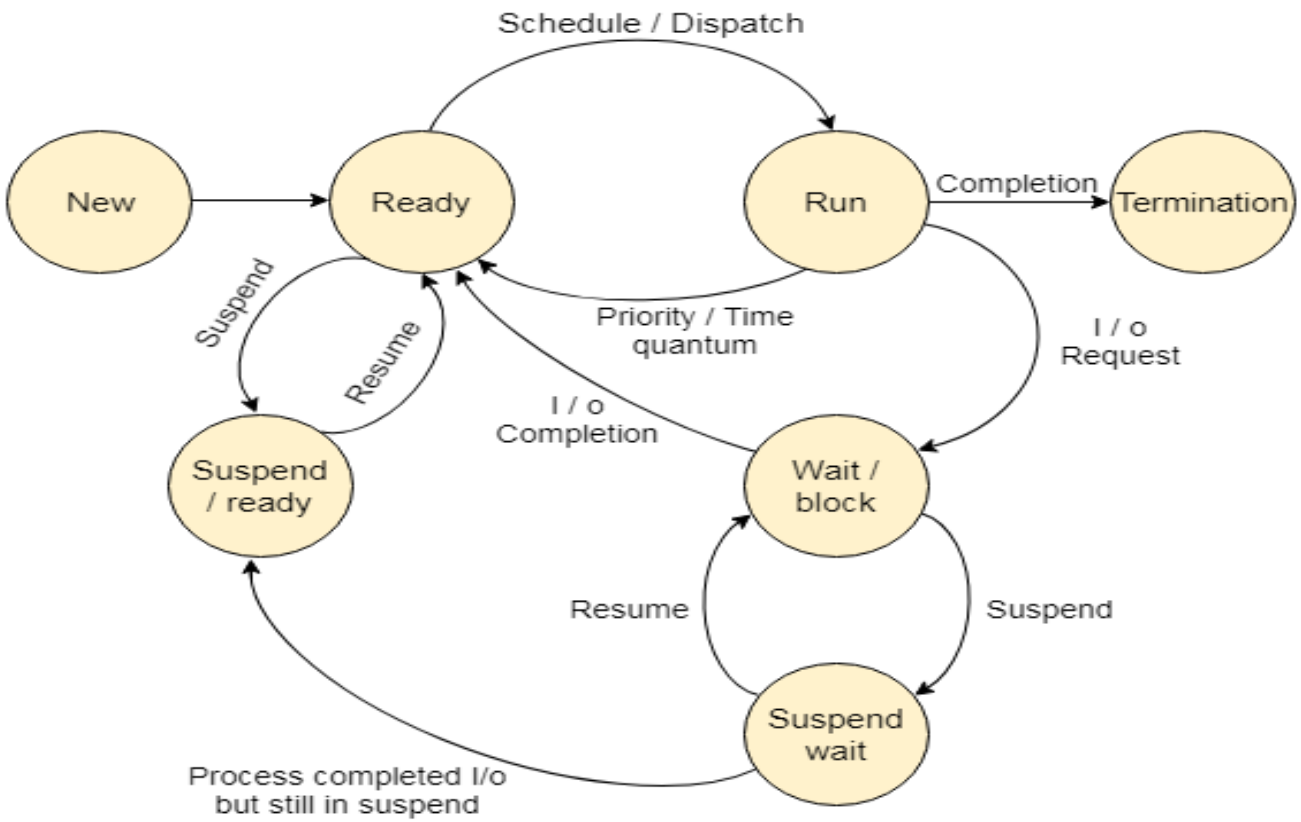
Process suspension

- Many OS are built around (Ready, Running, Blocked) states. But there is one more state that may aid in the operation of an OS - suspended state.
- When none of the processes occupying the main memory is in a Ready state, OS swaps one of the blocked processes out onto to the Suspend queue.

Suspend ready: A process in the ready state, which is moved to secondary memory from the main memory due to lack of the resources (mainly primary memory) is called in the suspend ready state.

Suspend wait: Instead of removing the process from the ready queue, it's better to remove the blocked process which is waiting for some resources in the main memory.

Process State Transition Diagram



Process Control Block(PCB)

- Each process is represented in the operating system by a process control block (PCB)-also called a task control block.
  - Process State - e.g. new, ready, running etc.
  - Process Number – Process ID
  - Program Counter - address of next instruction to be executed

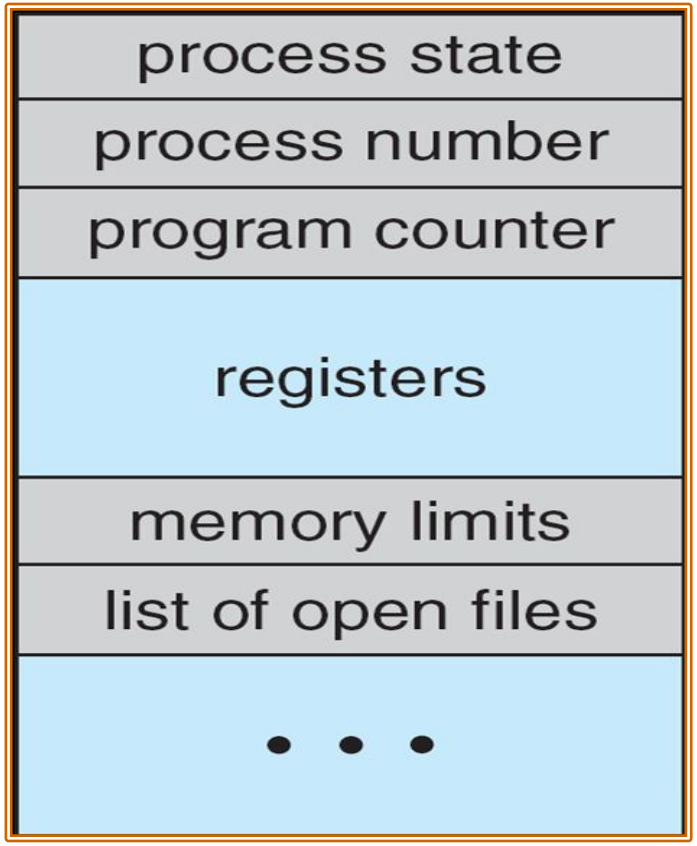


Figure 3.3. Process control block (PCB).

- CPU registers - general purpose registers, index registers, stack pointer etc.
- CPU scheduling information - This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
- Memory Management information - This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system.
- Accounting information - This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.
- I/O Status information - This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

Process Creation

Events which cause process creation:

- Initialization of operating system
- A user request to create a new process
- Initiation of a batch job
- Parent process create children process
- Address space
- Resource sharing
- Parent and child share no resources

Process Hierarchy

- Modern general purpose operating system used to create and destroy processes. A process may create several new processes during its time of execution.
- The creating process is called "Parent Process", while new processes are called "Child Processes".

There are different possibilities concerning creating new processes:-

Execution: The parent process continues to execute concurrently with its children processes or it waits until all of its children processes have terminated (sequentially).

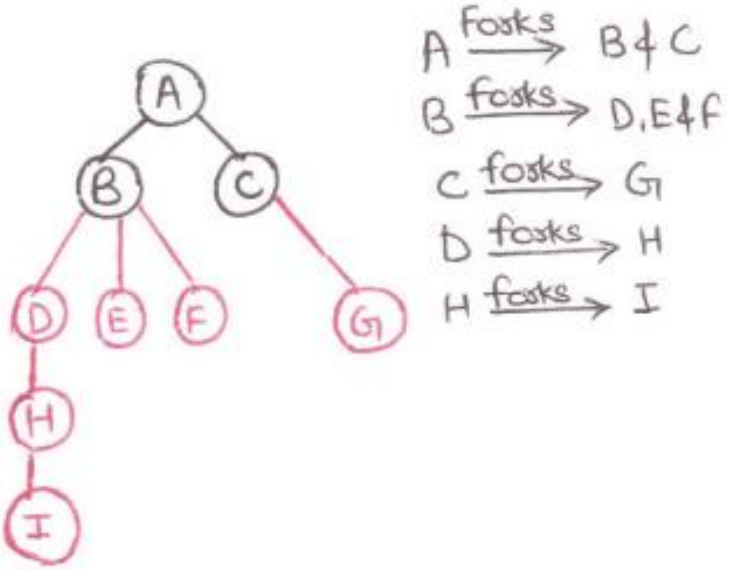
Sharing: Either the parent and children processes share all resources (like memory or files) or the children processes share only a subset of their parent's resources or the parent and children process share number of resources in common.

A parent process can terminate equation of one of its children for one of these reasons:

• The child process has exceeded its usage of the resources it has been allocated. For this a mechanism must be available to allow the parent process inspect the state of its children process.

• Task assigned to child process is no longer required.

In unix this is done by the **'Fork'** system call, which creates a **'child'** process and the **'exit system call'**, which terminates current process.



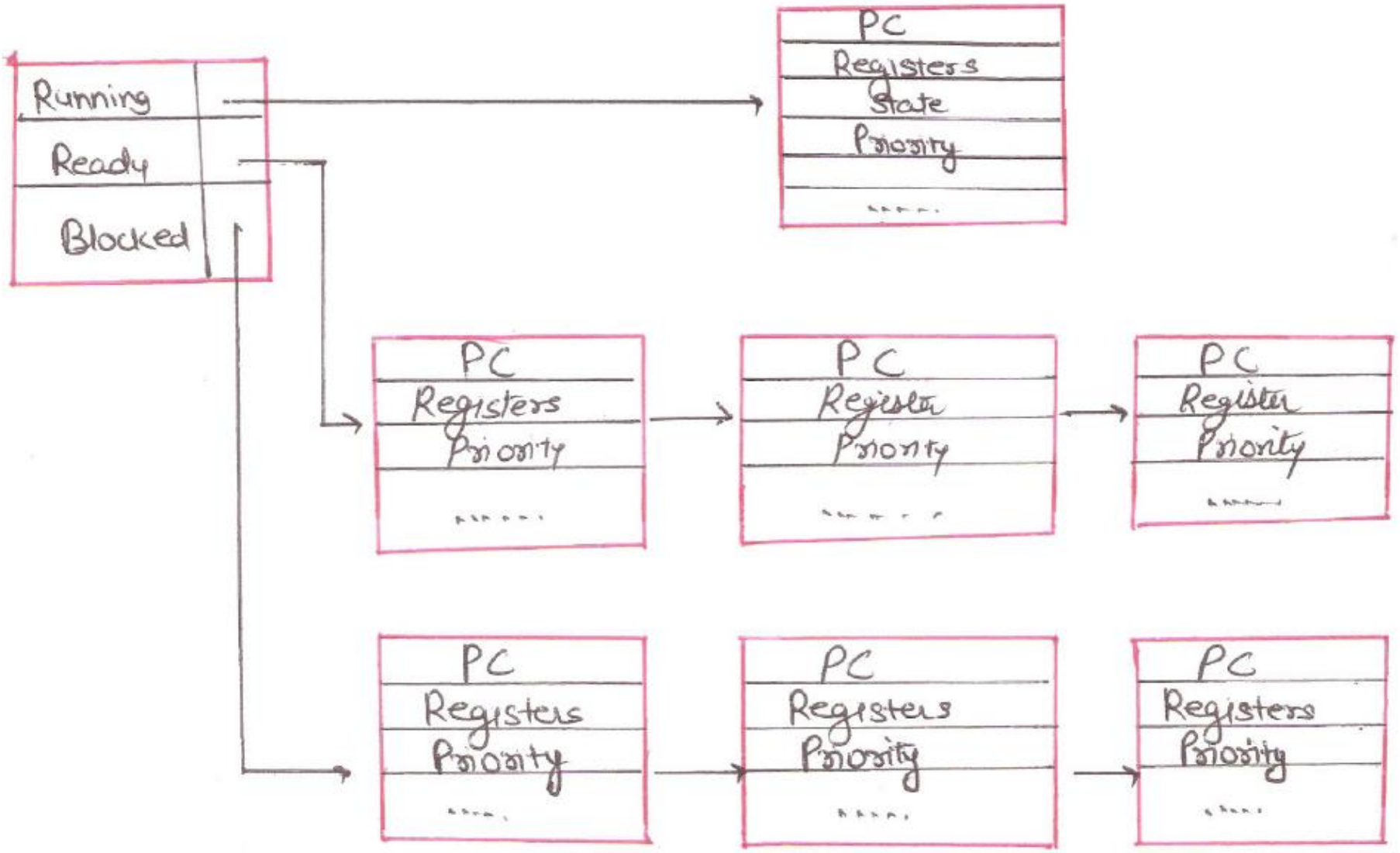
Process Termination

Events which cause process termination:

- Normal exit (voluntary).
- Error exit (voluntary).
- Fatal error (involuntary).
- Killed by another process (involuntary).
- When a process is terminated, the resources that were being utilized by the process are released by the operating system.
- When a child process terminates, it sends the status information back to the parent process before terminating.

Implementation of Processes

- Process Model is implemented by **Process Table and Process Control Block** which keep track all information of process.
- At the time of creation of a new process, operating system allocates a memory for it loads a process code in the allocated memory and setup data space for it .
- The state of process is stored as ' new ' in its PCB and when this process move to ready state its state is also changes in PCB.
- When a running process needs to wait for an input output devices, its state is changed to 'blocked'. The various queues used for this which is implemented as linked list.


Running
Ready
Blocked
PC
Registers
State
Priority
PC
Registers
Priority
PC
Register
Priority
PC
Register
Priority
PC
Registers
Priority
PC
Registers
Priority
PC
Registers
Priority

There are many following queues:

- **Read Queue:** This queue is used for storing the processes with state ready .
- **Blocked Queue:** it is used for storing the processes but need to wait for an input output device or resource.
- **Suspended Queue:** It is used for storing the blocked process that have been suspended.
- **Free process Queue:** it is used for the information of empty space in the memory where a new PCB can be created.

Process management	Memory management	File management
Registers	Pointer to text segment info	Root directory
Program counter	Pointer to data segment info	Working directory
Program status word	Pointer to stack segment info	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Figure. Some of the fields of a typical process-table entry.

Implementation of Processes

1. Hardware stacks program counter, etc.
2. Hardware loads new program counter from interrupt vector.
3. Assembly language procedure saves registers.
4. Assembly language procedure sets up new stack.
5. C interrupt service runs (typically reads and buffers input).
6. Scheduler decides which process is to run next.
7. C procedure returns to the assembly code.
8. Assembly language procedure starts up new current process.

Figure 3-5. Skeleton of what the lowest level of the operating system does when an interrupt occurs.