Operating System

Dr. Satyabrata Das

Associate Professor

Dept. of IT, VSSUT, Burla

Operating System

Definition:

Operating system is a system-software. By the use of system software we can bring the hard ware into a working environment so that the user can work in a systematic and reliable manner. The operating system controls the way software uses hardware. The purpose of this control is to make the computer operate in the way intended by the user, and in a systematic reliable and efficient manner. This view of the operating system is shown below

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

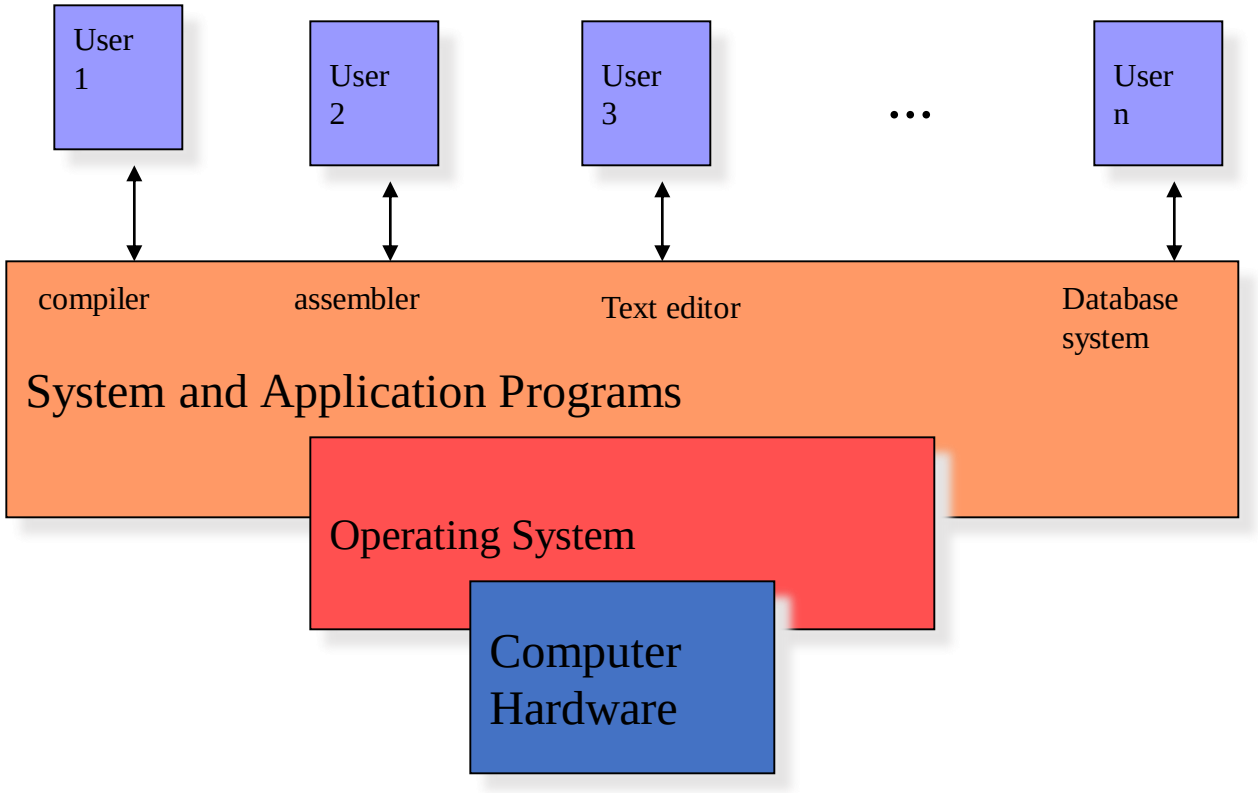
Computer System Structure

Computer system can be divided into four components:

- Hardware – provides basic computing resources
  - CPU, memory, I/O devices
- Operating system
  - Controls and coordinates use of hardware among various applications and users
- Application programs – define the ways in which the system resources are used to solve the computing problems of the users
  - Word processor, Apps, video games, Univ. Mgmt System etc
- Users
  - People, machines, other computers

Abstract View of System

Four Components of a Computer System



Functions of Operating system

Operating system is a set of system programs which provides an environment to help the user to execute the programs. Operating system is a resources manager which allocates and manages various resources like processor(s), main memory, i/o devices, information (files). The purpose of studying operating system concept is to understand the capabilities and the limitations of this resources manager, and understanding of the same helps in writing effective programs. Its several functions are given below:

(1)Command Interpreter

Every command that is typed by the user is interpreted by the operating system. After the interpreting, action is taken according to the command.

For example when the user types date, the system interprets it and prints the date and time in the terminal.

(2) Time Sharing

This is the way to allow more than one user to use the system at a time. While many users are using the system/computer simultaneously the operating system ensures that their job don’t get mixed with each other.

Functions of Operating system Contd...

(3)Security

The operating system protects one user from another user. No user is allowed to see the information manipulated by another user.

(4) Programme Development Tools

The tools are generally programs written by the system administrator and exports. They help the user to develop their own program in a better and neater and easier fashion. The collection of such tools provides an environment in which an ordinary user feels more comfortable.

(5) Communication Facilities

Operating system also provides the facilities to communicate with each other. These days interlinked technically, we say networks are build up for communication. Using these facilities operating system communicate with each other and let their user also communicate.

Functions of Operating system Contd...

(6)Input / Output

Through mouse, keybo**ard and** many other devices the user communicate with the system such as disk, tape, through which the information is supplied to (input) and obtained from (output) the computer. The operating system ensures smooth functioning of all these peripherals.

(7) Information Management and accounting

Every user of computer system wants to manipulate some kind of information. The operating system provides ways to store and manipulate this information. The operating system facilitates the collection of various statistics that help in improving performance.

Operating System

Dr. Satyabrata Das

Associate Professor

Dept. of IT, VSSUT, Burla

Components of Operating System

The four components of operating system are

- memory management
- device management
- process management
- information management.

All these resources are valuable and it is the function of operating system to see that they are used efficiently to resolve conflicts arising from competition among the various users.

The operating system must keep track of states of each resource, decide which process is to get the resource, allocate it and eventually reclaim it.

Functions 4 components

The measure functions of all these 4 resources are shown below:

(1)Memory Management

(a) Keep track of the resources/memory.

(b)In a multi programming environment it decides which job has priority/chance to get the resources and for how much time.

(c)Allocate the resources (memory) when the process requests it.

(d)Reclaim the resources (memory) when the process no longer needs it.

(2)Process Management

(a)Keep track of the resources (processor and status of the processor).

(b)Decide who will have a chance to use the processor.

(c)Allocate the resource (processor to a process by setting of the necessary hardware register).

(d)Reclaim the resources when the process surrenders that processor uses.

Functions 4 components Contd…

(3)Device management

(a)Keep track of the resources (devices). This is called the I/O traffic controller.
(b)Decides an efficient way to allocate the resources (devices).
(c)Allocate the resources (device) and initiate the I/O operation.
(d)Reclaim the resources.
.

(4)Information Management

(a)Keep track of the resource (information).
(b)Decide who to get use of the resources.
(c)Allocate the resources i.e., open a file.
(d)De-allocate the resources i.e., close a file.
.

Operating System Views

- Resource allocator
  - to allocate resources (software and hardware) of the computer system and manage them efficiently.
- Control program
  - Controls execution of user programs and operation of I/O devices.
- Kernel
  - The program that executes forever (everything else is an application with respect to the kernel).

Operating system roles

- **Referee**
  - Resource allocation among users, applications
  - Isolation of different users, applications from each other
  - Communication between users, applications
- **Illusionist**
  - Each application appears to have the entire machine to itself
  - Infinite number of processors, (near) infinite amount of memory, reliable storage, reliable network transport
- **Glue**
  - Libraries, user interface widgets, …
  - Reduces cost of developing software

Goals of an Operating System

- Simplify the execution of user programs and make solving user problems easier.
- Use computer hardware efficiently.
  - Allow sharing of hardware and software resources.
- Make application software portable and versatile.
- Provide isolation, security and protection among user programs.
- Improve overall system reliability
    - error confinement, fault tolerance, reconfiguration.

Why should I study Operating Systems?

- Need to understand interaction between the hardware and applications
  - New applications, new hardware..
  - Inherent aspect of society today
- Need to understand basic principles in the design of computer systems
  - efficient resource management, security, flexibility
- Increasing need for specialized operating systems
  - e.g. embedded operating systems for devices - cell phones, sensors and controllers
  - real-time operating systems - aircraft control, multimedia services

The OS is Everywhere

```
main(int argc, char **argv)
{ int fd = open(argv[1], O_RDONLY);
   if (fd < 0) {
     fprintf(stderr, "Failed to open\n");
     exit(-1);
   }
   while (1) {
     if (read(fd, &c, sizeof c) != 1)
       exit(-1);
    putc( c )
  }
}
```

- Edit
- Compile
- Run/Create Process
- Invoke main
- Open file
  - check access
  - cache
  - read character
- Write character
- Terminate process on EOF or Err

% cc main.c
% ./a.out /tmp/foo.bar

Operating System

Dr. Satyabrata Das

Associate Professor

Dept. of IT, VSSUT, Burla

Major issues in Operating Systems

- **structure -- how is an operating system organized?**
- **sharing -- how are resources shared among users**
- **naming -- how are resources named (by users or programs)**
- **protection -- how is one user/program protected from another**
- **security -- how to restrict the flow of information**
- **performance -- why is it so slow?**
- **reliability and fault tolerance -- when something goes wrong**
- **extensibility -- how do we add new features?**
- **communication -- how and with whom can we communicate (exchange information)**

Major issues in OS (2)

- **concurrency -- how are parallel activities created and controlled?**
- **scale and growth -- what happens as demands or resources increase?**
- **persistence -- how to make data last longer than programs**
- **compatibility -- can we ever do anything new?**
- **distribution -- accessing the world of information**
- **accounting -- who pays the bills, and how do we control resource usage?**

A brief history of operating systems

- **“in the beginning”, the OS was just code to which you linked your program, loaded the whole thing into memory, and ran your program; basically, just a run-time library**
- **simple batch systems were first real operating systems:**
  - os was stored in part of primary memory
  - it loaded a single job (from card reader) into memory
  - ran that job (printed its output, etc.)
  - loaded the next job...
  - *control cards* in the input file told the os what to do
- ***Spooling* and *buffering* allowed jobs to be read ahead of time onto tape/disk or into memory.**

Early Systems - Bare Machine (1950s)

Hardware – *expensive* ; Human – *cheap*

- Structure
  - Large machines run from console
  - Single user system
    - Programmer/User as operator
  - Paper tape or punched cards
- Early software
  - Assemblers, compilers, linkers, loaders, device drivers, libraries of common subroutines.
- Secure execution
- Inefficient use of expensive resources
  - Low CPU utilization, high setup time.

From John Ousterhout slides

Simple Batch Systems (1960's)

- Reduce setup time by batching jobs with similar requirements.
- Add a card reader, Hire an operator
  - User is NOT the operator
  - Automatic job sequencing
    - Forms a rudimentary OS.
  - Resident Monitor
    - Holds initial control, control transfers to job
  - Problem
    - Need to distinguish job from job and data from program.

From John Ousterhout slides

Supervisor/Operator Control

- Secure monitor that controls job processing
  - Special cards indicate what to do.
  - User program prevented from performing I/O
- Separate user from computer
  - User submits card deck
  - cards put on tape
  - tape processed by operator
  - output written to tape
  - tape printed on printer
- Problems
  - Long turnaround time - up to 2 DAYS!!!
  - Low CPU utilization
    - I/O and CPU could not overlap; slow mechanical devices.

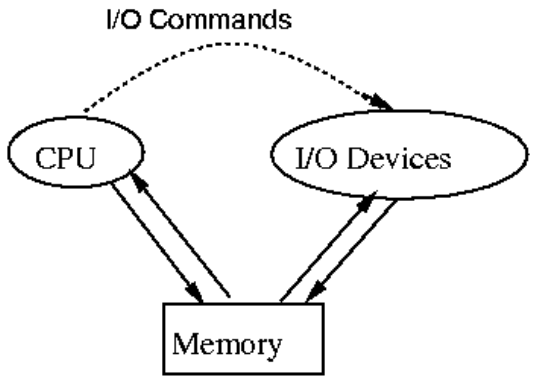
From John Ousterhout slides

Batch Systems - Issues

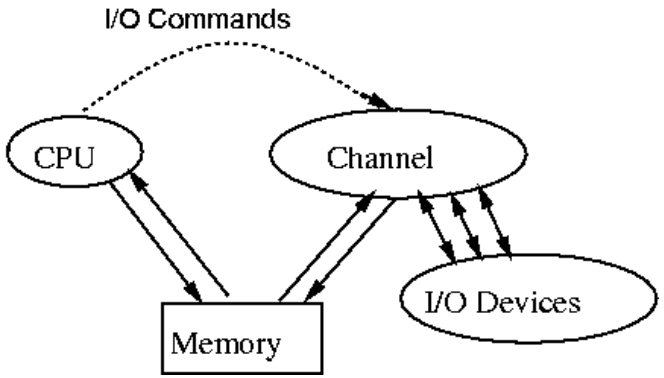
- Solutions to speed up I/O:
- Offline Processing
  - load jobs into memory from tapes, card reading and line printing are done offline.
- Spooling
  - Use disk (random access device) as large storage for reading as many input files as possible and storing output files until output devices are ready to accept them.
  - Allows overlap - I/O of one job with computation of another.
  - Introduces notion of a job pool that allows OS choose next job to run so as to increase CPU utilization.

Speeding up I/O

- Direct Memory Access (DMA)



- Channels



Multiprogramming

- *Multiprogramming* systems provided increased utilization
  - keeps multiple runnable jobs loaded in memory
  - overlaps I/O processing of a job with computes of another
  - benefits from I/O devices that can operate asynchronously
  - requires the use of interrupts and DMA
  - tries to optimize *throughput* at the cost of response time

Timesharing

- ***Timesharing*** **supported *interactive* use of**
  - each user feels as if he/she has the entire machine (at least late at night!)
  - timesharing tries to optimize *response time* at the cost of throughput
  - based on time-slicing -- dividing CPU equally among the users
  - permitted active viewing, editing, debugging, participation of users in the execution process
- **MIT Multics system (mid-late 1960s) was first large timesharing system**

Personal Computing Systems

Hardware – *cheap* ; Human – *expensive*

- Single user systems, portable.
- I/O devices - keyboards, mice, display screens, small printers.
- Laptops and palmtops, Smart cards, Wireless devices.
- Single user systems may not need advanced CPU utilization or protection features.
- Advantages:
  - user convenience, responsiveness, ubiquitous

Distributed Operating Systems

- **distributed systems facilitate use of geographically distributed resources**
  - **machines connected by wires**
- **supports communication between parts of a job or different jobs**
  - **interprocess communication**
- **sharing of distributed resources, hardware and software**
  - **resource utilization and access**
- **permits some parallelism, but speedup is not the issue**

Parallel Operating Systems

- Support parallel applications wishing to get speedup of computationally complex tasks
- Needs basic primitives for dividing one task into multiple parallel activities
- Supports efficient communication between those activities
- Supports synchronization of activities to coordinate sharing of information
- It’s common now to use networks of high-performance PCs/workstations as a parallel computer

Embedded Operating Systems

- The decreased cost of processing makes computers ubiquitous. Each "embedded" application needs its own OS or control software:
  - cell phones
  - PDAs (Palm Pilot, etc.)
  - "network terminals" (internet interfaces)
- In the future, your house will have 100s of these things in it (if it doesn't already)

Real-time systems

- Correct system function depends on timeliness
- Feedback/control loops
- Sensors and actuators
- Hard real-time systems -
  - Failure if response time too long.
  - Secondary storage is limited
- Soft real-time systems -
  - Less accurate if response time is too
  - Useful in applications such as multimedia, virtual reality.

A personal computer today

interaction

- Super AMOLED display
- Capacitive touchscreen (multitouch)
- Audio (speaker, microphone)
- Vibration
- S pen

- 4G LTE
- NFC
- WiFi
- Bluetooth
- Infrared

- 64 GB internal storage (extended by microSD)
- Adreno 330 GPU
- Hexagon DSP
- Multimedia processor

- 13 MP front camera
- 2 MP back camera
- Accelerometer
- Gyroscope
- Proximity sensor
- Compass
- Barometer
- Temperature sensor
- Humidity sensor
- Gesture Sensor
- GPS

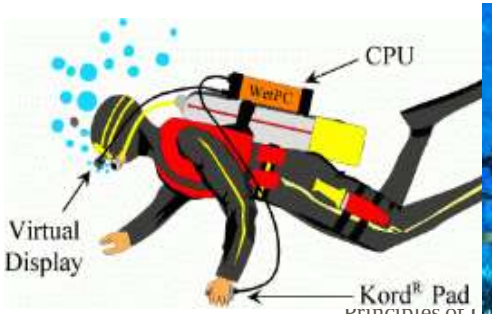
Operating systems are everywhere

Operating systems are everywhere






Systems Today



Irvine Sensorium

Operating System

Dr. Satyabrata Das

Associate Professor

Dept. of IT, VSSUT, Burla

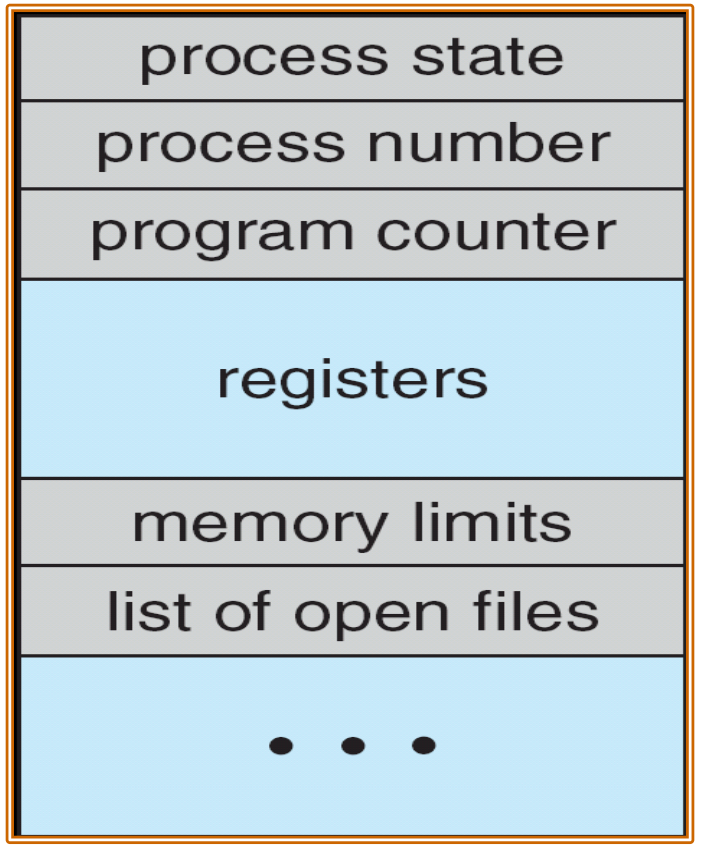
Process Creation

When a process executes, it changes the state, generally the state of process is determined by the current activity of the process. Each process may be in one of the following states

- **New** - The process is being created.
- **Running** - Instructions are being executed.
- **Waiting** - Waiting for some event to occur.
- **Ready** - Waiting to be assigned to a processor.
- **Terminated** - Process has finished execution.

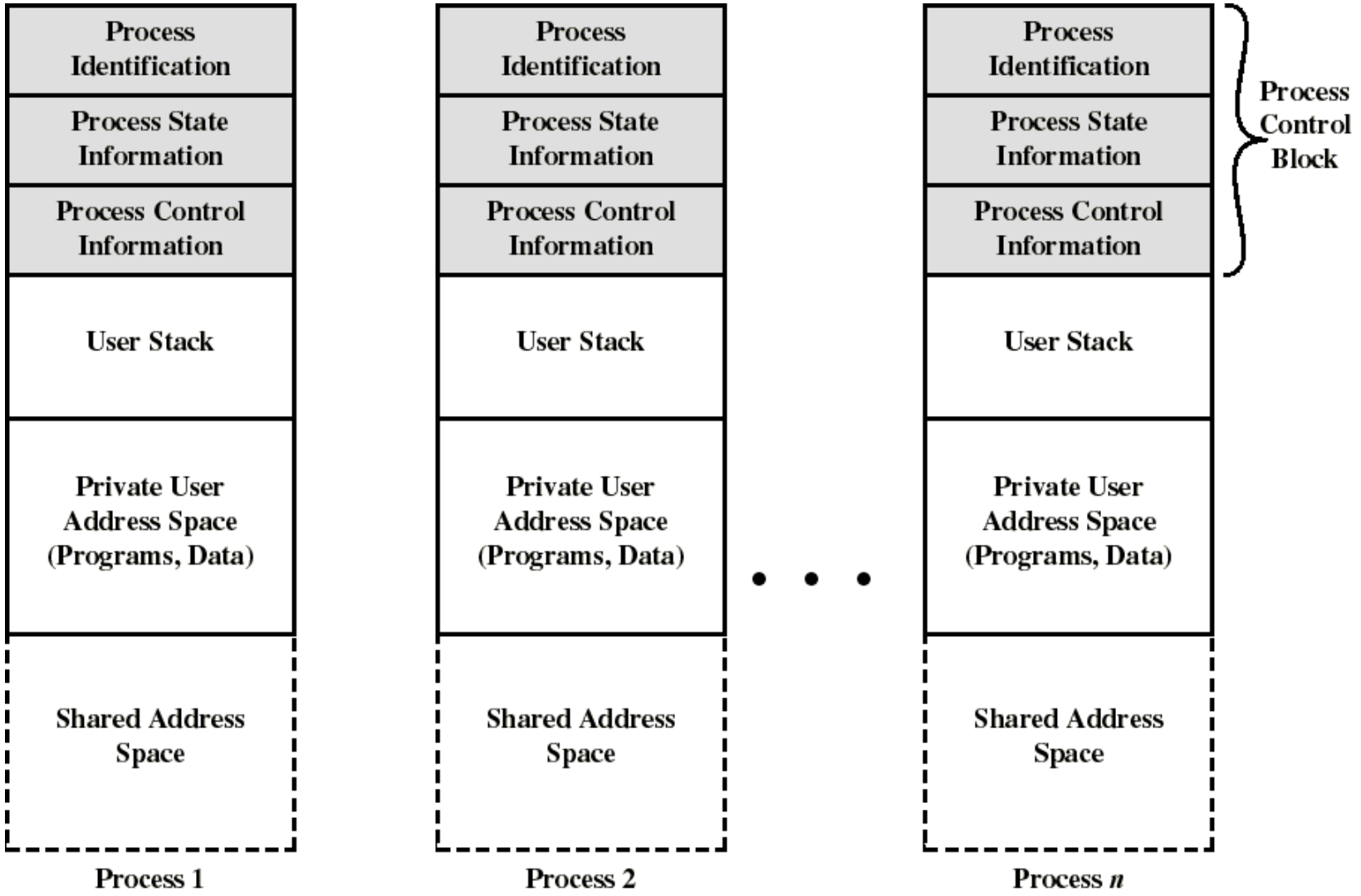
Process Control Block

- Contains information associated with each process
  - Process State - e.g. new, ready, running etc.
  - Process Number – Process ID
  - Program Counter - address of next instruction to be executed
  - CPU registers - general purpose registers, stack pointer etc.
  - CPU scheduling information - process priority, pointer
  - Memory Management information - base/limit information
  - Accounting information - time limits, process number
  - I/O Status information - list of I/O devices allocated



Process Control Block

Process images in virtual 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

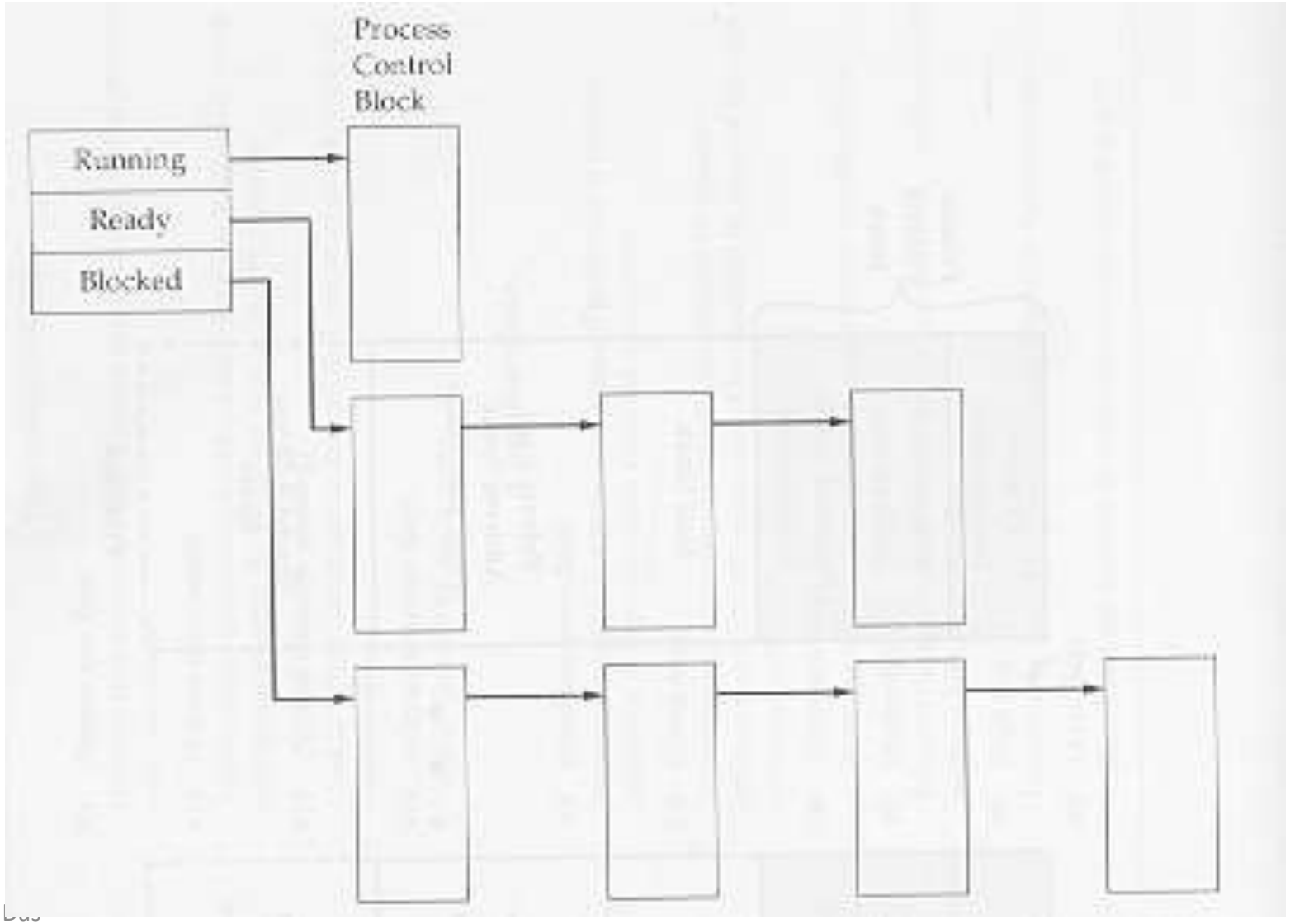
Processor State Information (in PCB)

- 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...)

Operating System

Dr. Satyabrata Das

Associate Professor

Dept. of IT, VSSUT, Burla

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

Reasons for Process Creation

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 logon	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 Creation

Process spawning	*Parent process*	*Child process*
• when the OS creates a process at the explicit request of another process	• is the original, creating, process	• is the new process

Process Termination

There must be a means for a process to indicate its completion

A batch job should include a HALT instruction or an explicit OS service call for termination

For an interactive application, the action of the user will indicate when the process is completed(e.g. log off, quitting an application)

Reasons for process termination

The process terminate from the running state includes so many causes. Generally the process terminates when execution finished. Some other causes are:

- Time slot expired
- Memory Violation
- I/O failure
- Parent termination
- Parent request
- Invalid Instruction

Operations on process

System that manage process must be able to perform certain operations and with process. These include

- Create a process
- Destroy a process
- Resume a process(restart the process)
- Change the priority of process
- Block of process
- Wakeup a process
- Dispatch a process
- Enable a process to communicate with other process

Operating System

Dr. Satyabrata Das

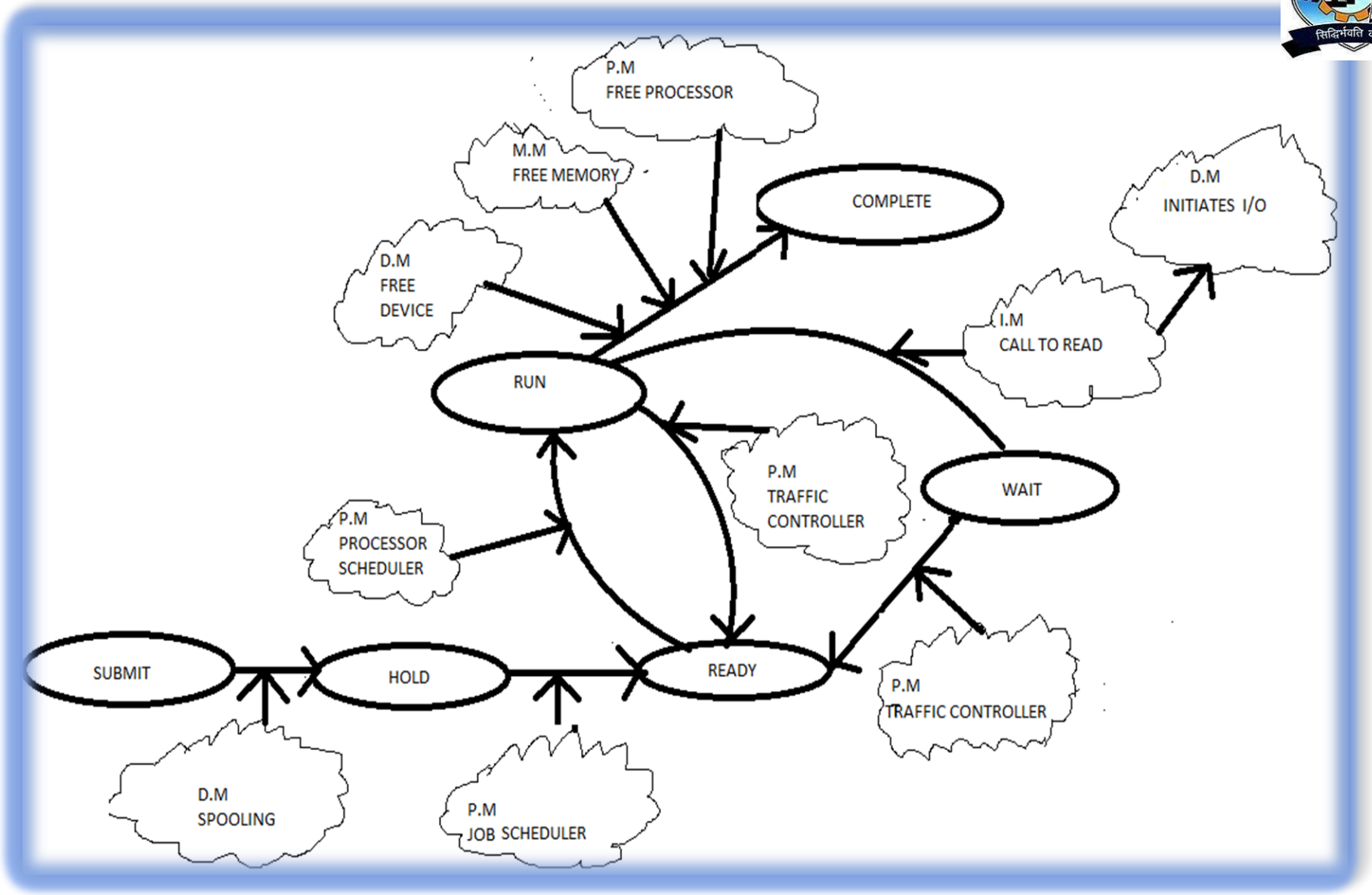
Associate Professor

Dept. of IT, VSSUT, Burla

Outlines

- Introduction to Process Life cycle
- Privileged Instructions
- Overlays
- Time sharing system
- Buffering
- Batch Processing
- SPOOLING
- Real-time System
- Process management

Process states/Life Cycle of a process



AN OPERATING SYSTEM-PROCESS VIEW POINT

Submit:A user submits a job to the system. The system must respond the user's request.
Hold: The user's job has been converted into internal machine readable form, but no resources have been assigned to the process.
Ready: The process is ready to run but there are more processes than processors and it must wait for its turn on processor.
Run: The process has been assigned a processor as its programs are presently being executed.
Wait: The process is waiting for some event (an I/O operation to be completed).
Complete: The process has completed its computation all its assigned resources may be reclaimed.

Description of process life cycle

- A user submits job to the system by placing a deck into a card reader (submit state). The job consists of several decks of programs predicate by job card controller. The job card controller passes information to the operating system as to what resources the job will need. The spooling routine reads the job and places it into a disk (HOLD). The job scheduler determines the device requirements from the user's job card. Once the job scheduler decides that the job has to be assigned resources the traffic controller is called to create the associated process of information and memory management is called to allocate the necessary main storage. The job is then loaded into memory and the process is ready to run (READY STATE). When a processor becomes free the process scheduler scans the list of the ready process, choose a process and assign a processor to it(RUNNING STATE). If the running process requests access to read a file, information management will call device management to initiate the process reading of the file (device management would initiate the reading of the file). Device management would initiate the I/O operation and then call the process management to indicate that the process requesting the file is awaiting the completion of the I/O operation (WAIT STATE).
- When the I/O operation is completed the traffic controller places the process back into the ready state. If the process completes its computation when it is run again then it will be placed into completed state and the allocated resources will be freed.

Description of process life cycle Contd…

(1)PREVILEGED INSTRUCTION

- It is a special type of instruction which is executed only in monitor mode. Operating system can execute the privileged instruction as it executes in monitor mode. So it changes the system state from user mode to the monitor mode.
- To provide better service we have two separate modes of operations i.e., user mode and monitor mode (also called system mode).
- A bit is added to the hard ware of the computer to indicate the current mode is user (1), monitor (0). I/O instruction and instruction to modify the memory management registers or the timer are privileged instructions. The h/w allows privileged instruction to be executed only in the monitor mode. If an attempt is made to execute privileged instructions in the user mode the h/w will not execute but it treats it as an illegal instruction and trap to the resident monitor. So when a trap occurs the h/w switches from user mode to monitor mode. Example=HALT instruction` is privileged, as a user program should never be able to halt the computer. The instructions to turn the interrupt system on and off are also privileged.

Description of process life cycle Contd…

(2)OVERLAYS

When a programmer used to write a large program that could not fit into the main memory. It is necessary to divide the program into small portions, so each one could fit into the primary memory. These small portions are called overlays. A programmer has to design overlays so they are independent of each other. Under these circumstances one can successively bring each overlay into the main memory and execute them in a sequence. But it increases the program development time considerably.

Description of process life cycle Contd…

(2)OVERLAYS

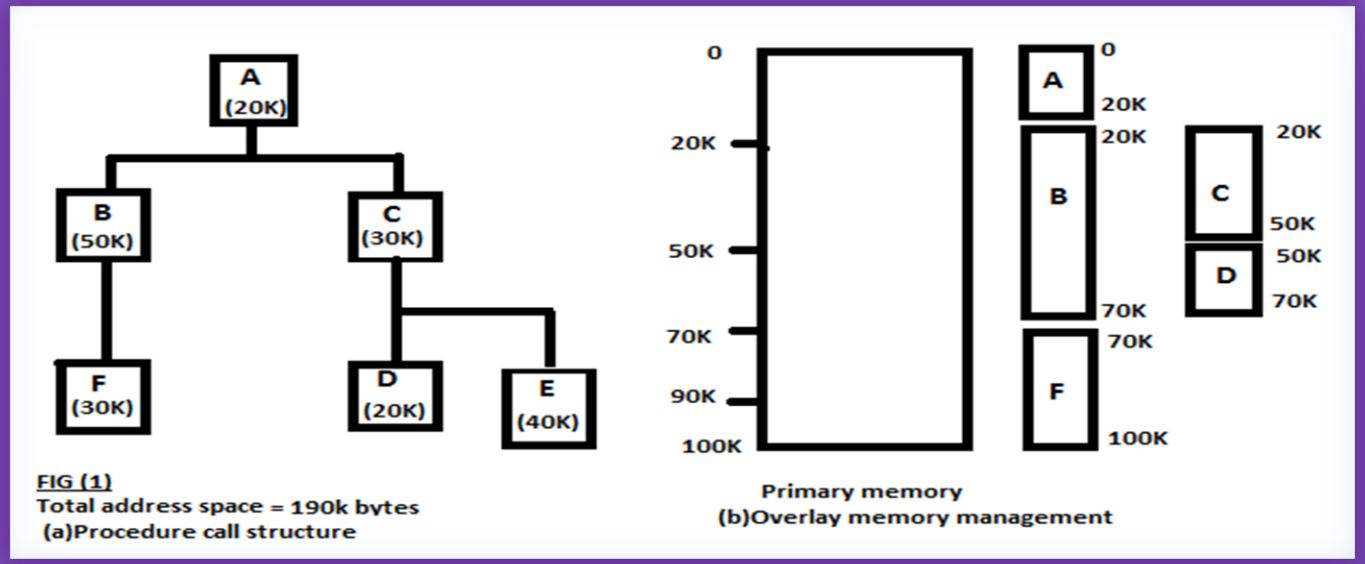


Fig (1) illustrates the relation between the procedures in a job's address space. This diagram indicate that procedure 'A' call only procedure 'B' and 'C', procedure 'B' call only procedure 'F', where as procedure 'C' call only procedure 'D' and 'E'. Thus procedure 'B' and 'C' do not have to be in memory at the same time. So instead of using 190k bytes to hold the job's address space in memory, maximum of 100k bytes are needed.

Description of process life cycle Contd…

(3)TIME SHAIRING SYSTEM

- Multiprogramming system that process many user programs concurrently in an interactive manner are called time sharing system. Here interactive manner means the computer and terminal user may communicate with each other.
- In such a system a clock is used to divide up processor time into “time slices” and these time slices are shared between users in an appropriate way. This is called time sharing. If a job is not completed at the end of its time slice, then it is interpreted and returned to the end of a queue to wait for another “time slice”. This policy guarantees fast response to user’s request, which can be processed with a single time slice.

Description of process life cycle Contd…

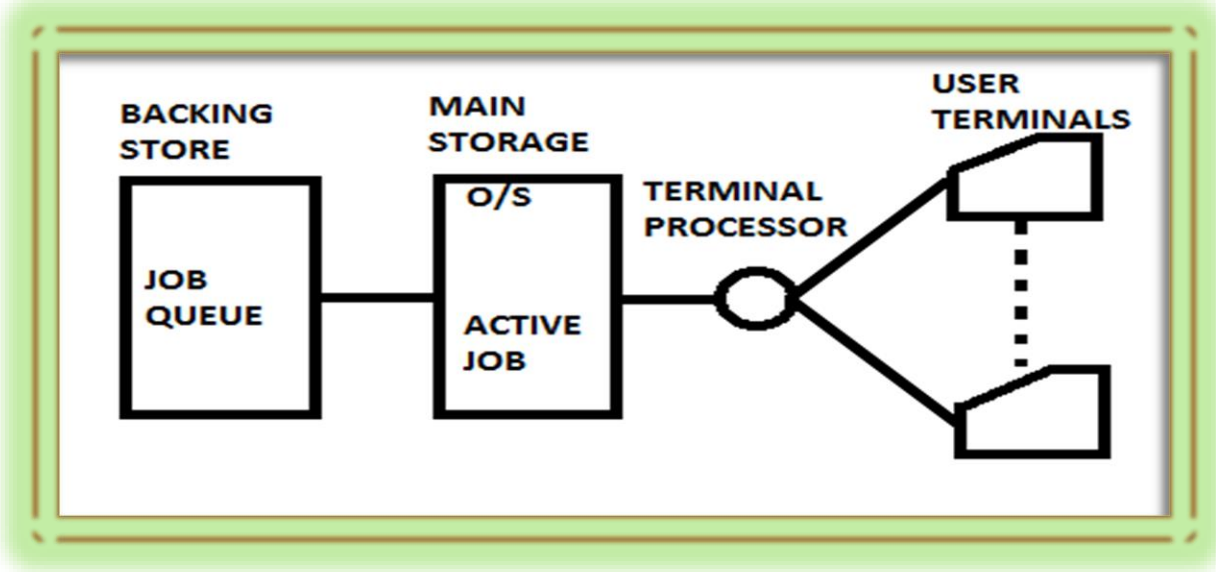
(3)TIME SHAIRING SYSTEM CONTD….

Example: let us take a system having following details

The average time slice is about 40 msec. at the end of this intervals, the active job is transferred to the drum and another job is loaded into the internal store. This exchange of jobs between two levels of storage is called swapping.

It takes roughly 40 msec. so provide fast response needs 80 ms for one job(i.e., one terminals)

Let us take that the system has 25 terminals so a user can expect response to a simple system request (with a single time slice only) in 25x80 msec = 2sec. so here we see that the time sharing system provide fast response to user request.



Description of process life cycle Contd…

(4)BUFFERING

- It is the name to the technique of transferring temporary storage tri processing thus enabling the simultaneous operations of the device can be achieved because of the autonomous operation of the peripheral which leaves the CPU free from other work. It performs.

  (*) Offline operations,

  (*) Computation of one program, and output of the same program.

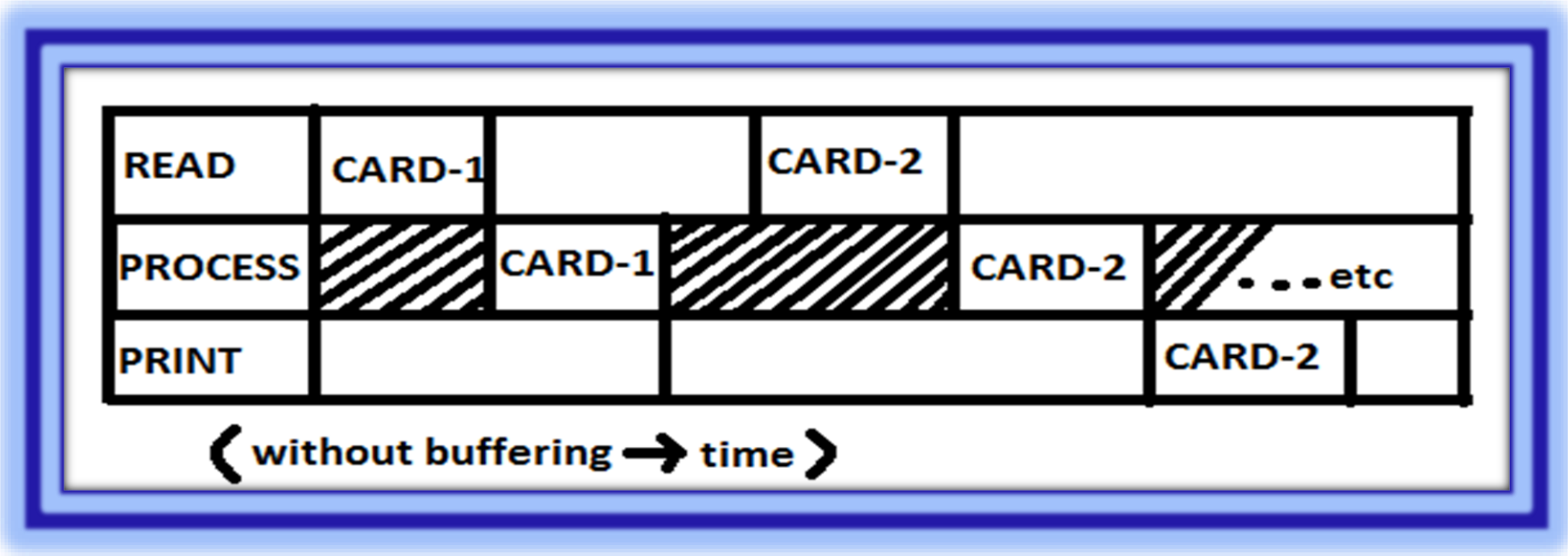
  Buffering keeps both the CPU and its I/O device busy all the time, it means if the CPU is working on one record while I/p device is working on another, either the CPU or I/P device will finish first. If the CPU finishes first then it must wait. It can't proceed until the second record is read. The CPU will remain idle for a long time. If the I/P device finishes first then either it must wait or it may proceed with reading the next record. So if the buffering technique is used then the I/P device can read several records ahead of the CPU.

Description of process life cycle Contd…

(4)BUFFERING

Example:

Suppose 2000 cards need to be input and the detail on the cards printed out on a line printer (one is being used for each card read). The card reader operates at 1000 cards per minute and the printer at 500 times per minute. The processing cycle and time required would be illustrated in fig (1), if buffering were not used and as illustrated in fig (2), if buffering were used.



Description of process life cycle Contd…

(4)BUFFERING

Here one activity takes place at a time i.e., as the card is processed the reade and printer idle.

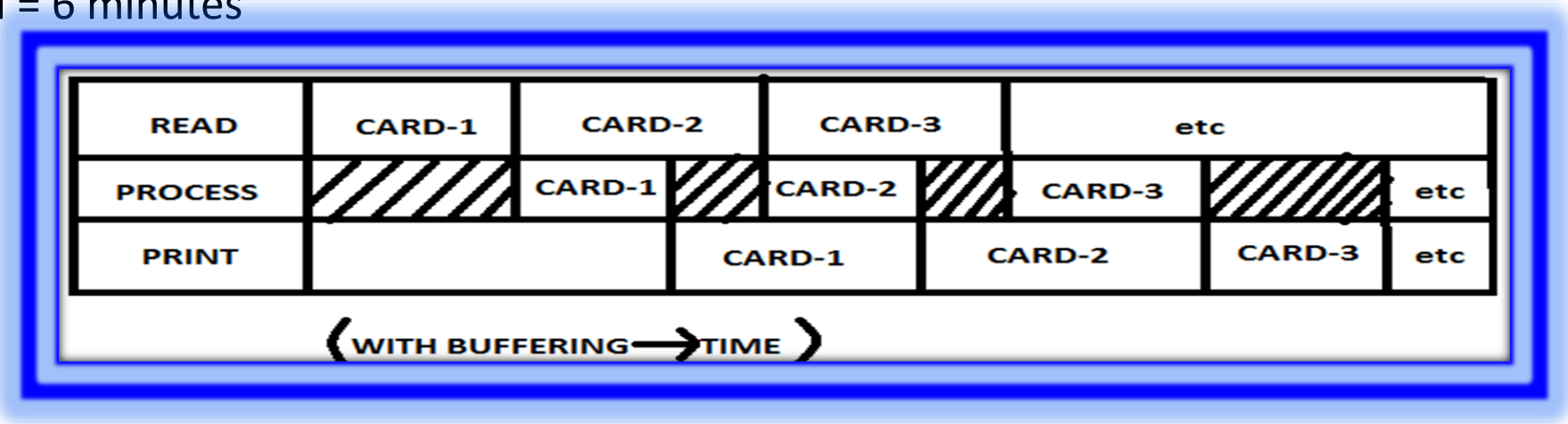
Total time required:-

Read = 2000 x (1/2000) = 2 min.

Write =negligible

Print = 2000 x (1/500) = 4 minutes

Total = 6 minutes



TOTAL TIME :After reading the first card all subsequence reading, processing and processing and printing can be overlapped . time required is a little over the speed of the device i.e., just over 4 minutes.

Operating System

Dr. Satyabrata Das

Associate Professor

Dept. of IT, VSSUT, Burla

Description of process life cycle Contd…

(5)BATCH PROCESSING

It is the execution of a series of programs (jobs) on a computer without manual interaction. Jobs are set up so that they can be run to completion without manual interaction. So all input data are preselected through script, command line parameter or job control language. This is in contrast to online or interactive program which prompt the user for such input. A program takes a set of data files as input, process the data and produces a set of output data files. This operating environment is termed as Fetch Processing because the input data collected into batches of files and are processed in batches by the program.

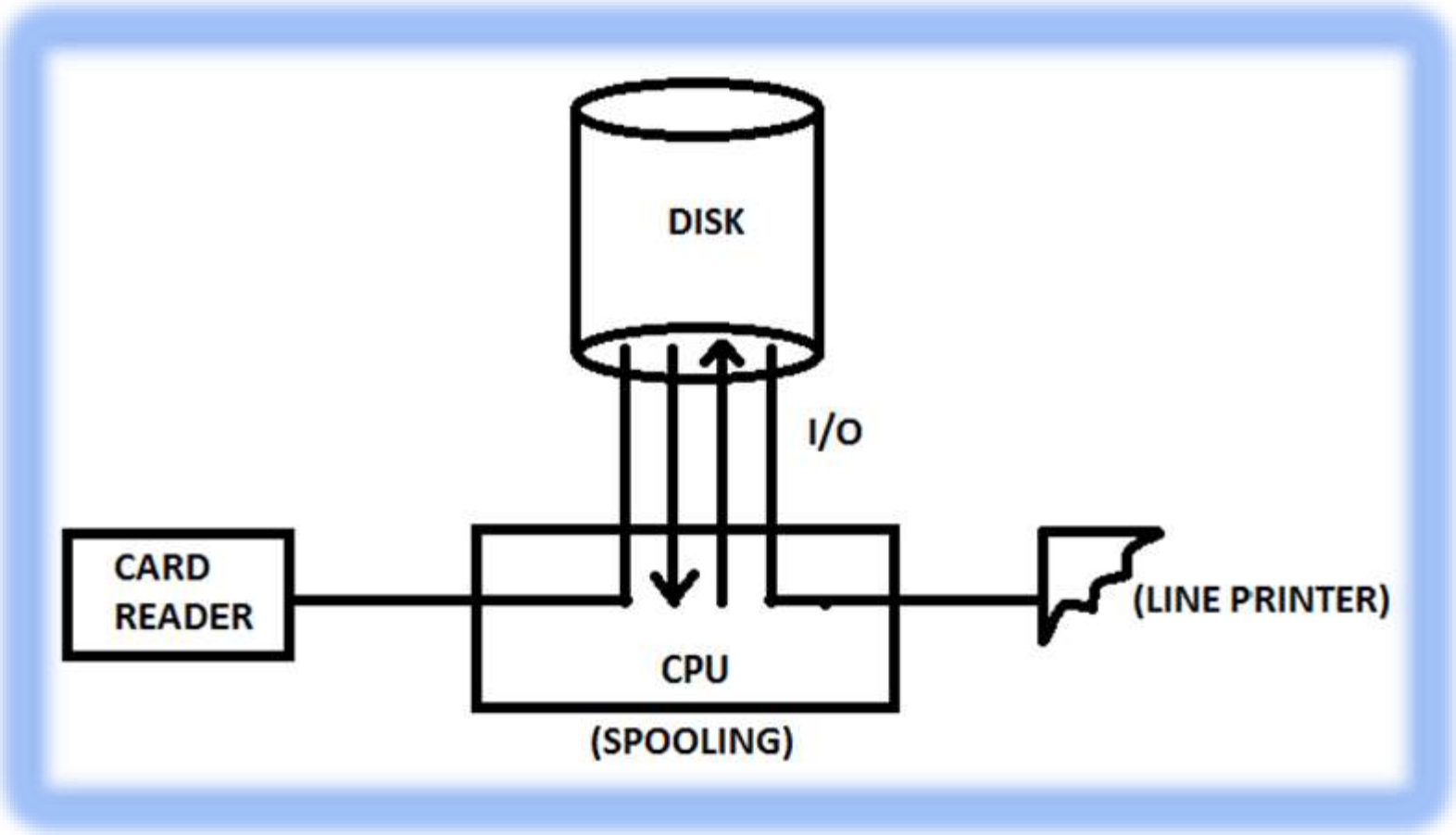
Description of process life cycle Contd…

(6)SPOOLING

- It is an acronym for simultaneous peripheral operation online. Spooling essentially uses the disk as a very large purpose for reading as far ahead as possible on input devices as for storing output files until the output devices are able to accept them. It performs:
- (*)Online operation
- (*)Computation of one program and output may be of another program.
- Spooling can be done with magnetic tape. It is possible to read the entire contents of the magnetic tape on to disk before using it. All operations then occur on the disk copy at higher speed and with no ware to the tape. This skim is known as staging a tape.

Description of process life cycle Contd…

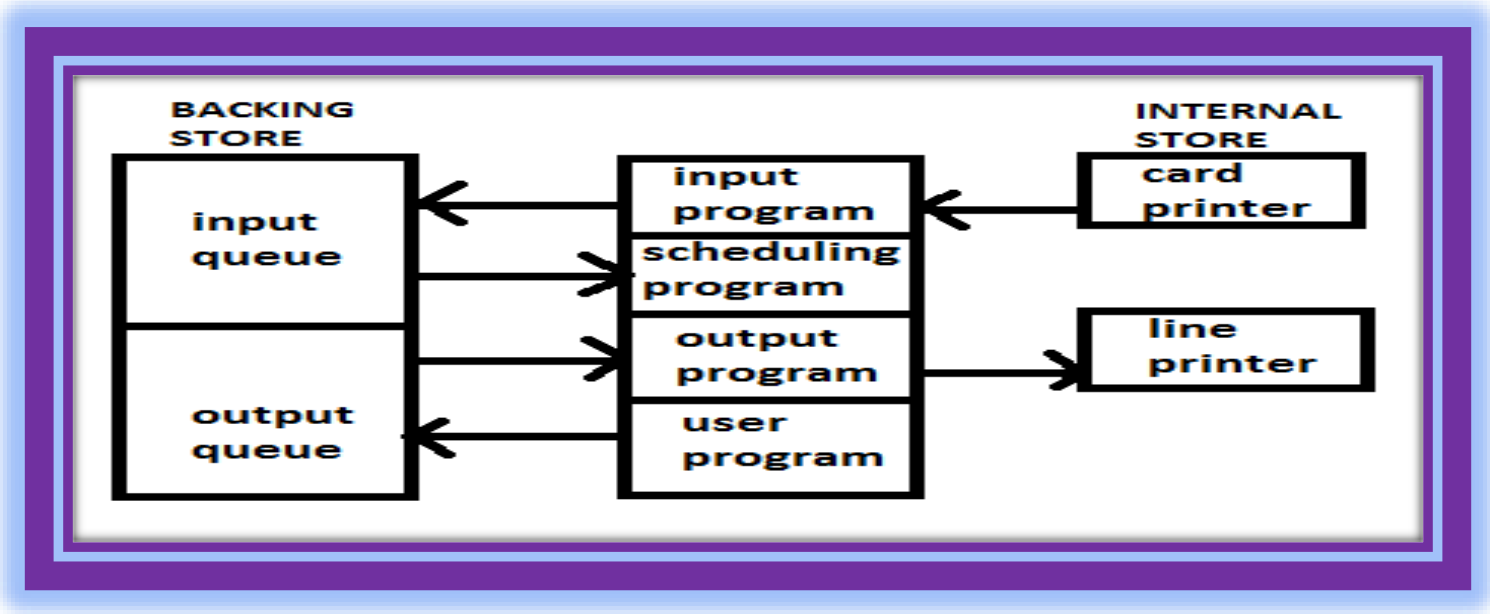
(6)SPOOLING CONTD…..



Description of process life cycle Contd…

(6)SPOOLING CONTD..

EXAMPLE: A very successful operating system with input or output spooling called EXEC-II was designed by computer science organization (Lynch, 1967 & 1971). It controlled a UNIVAC 1107 computer with an instruction execution time of 4sec. the backing store consisted of two or more fast drums, each capable of transferring 10,000 characters during a single revolution of 33 msec.



Description of process life cycle Contd…

(7)REALTIME SYSTEM

In a real time system the time at which output is produced is significant. This is usually because the input corresponds to some movement in the physical world and the output has to relate to that same movement.

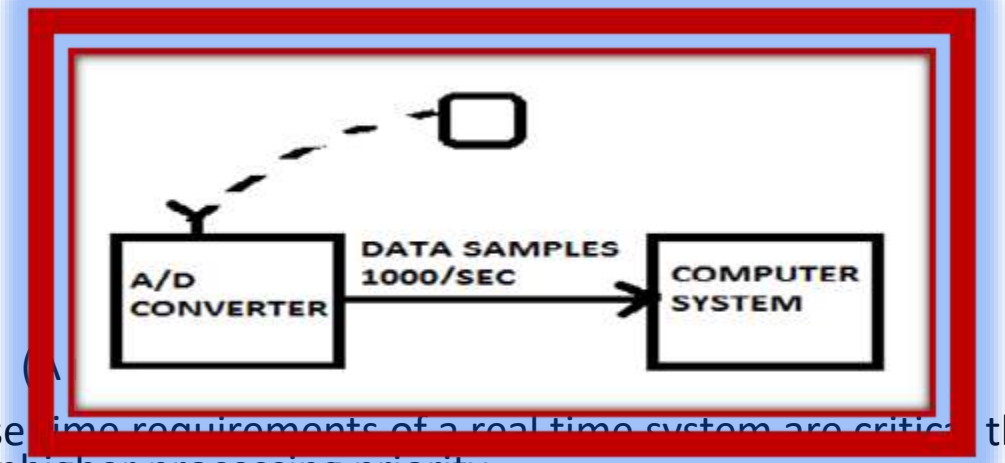
The main advantages of real time system is that the response time is of order milliseconds

Or

In the real time system, digitized samples are generated at the rate of 1000 samples per second. The computer is only required to store these samples in a specified file. Since a new sample comes in every in every “store the sample” request in less than 1ms. If it fails to do so, a sample would be lost and the real time application would have malfunctioned. The worst case response time is 1ms.

Description of process life cycle Contd…

(7)REALTIME SYSTEM CONTD…



Since the response time requirements of a real time system are critical the real time application would be given nhigher processing priority.

Note

In areal time system the response time is of order millisecond.

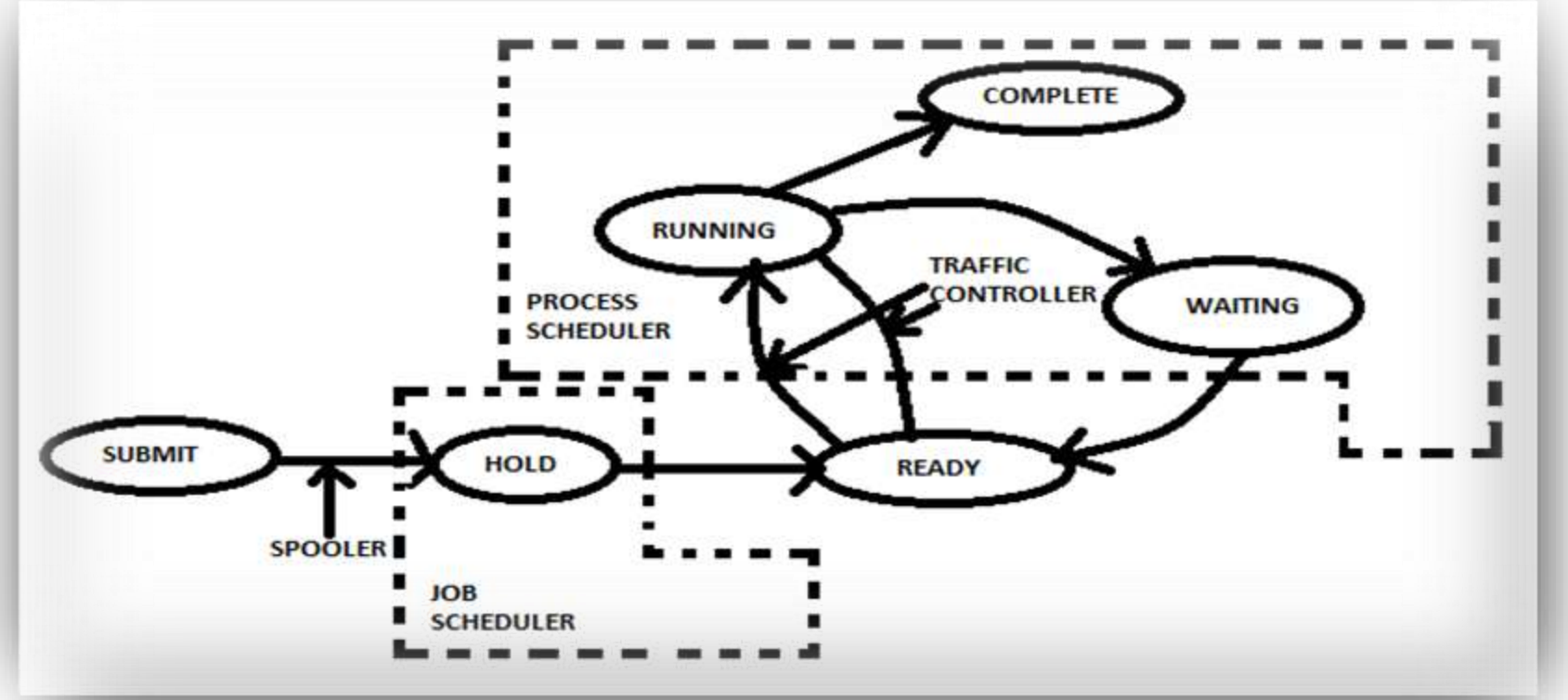
In an interactive system the response time is of order second.

In a batch system the response time is of order hour.

Description of process life cycle Contd…

(8)PROCESS MANAGEMENT

The fundamental function of the processor management component is to organize the execution of user jobs on the central processor



STATE MODEL

Description of process life cycle Contd…

(7)PROCESS MANAGEMENT CONTD..

The following are the modules of processor management

SPOOLER

Place all submitted jobs into a form for processing by the job scheduler.

JOB SCHEDULER

- Keeps tracks of all the jobs in the system.
- Select a job to run and creates corresponding process.

PROCESSING SCHEDULING

- Select a process to run.
- Allocate a processor.

TRAFFIC CONTROLER

- Keep track of the status of all processes.
- Provide the mechanics of changing process states.
- Co-ordinate inter-process synchronization and communication.

Description of process life cycle Contd…

(7)PROCESS MANAGEMENT CONTD..

JOB SCHEDULER

It is a program module which keeps track of all the available jobs. It uses a policy that which job is to be selected and when to be selected in a non multiprogramming environment.

PROCESS SCHEDULER

It keeps track of all available jobs. It uses a policy that which of the ready processes receives a processor at what time and how long in a multiprogrammed environment.

Description of process life cycle Contd…

(7)PROCESS MANAGEMENT CONTD..

JOB SCHEDULING

The job scheduling algorithm of a batch system takes into account not only the time a job arrives but also priority, memory needs, device needs, processor needs and system balance.

One mechanism for keeping track of jobs is to have a separate job control block (JCB) for each job in the system. When a process is placed in hold state, JCB is created for it with entries regarding its status and position in job queue.

The job scheduling process does these steps as below:

Keep track of status of all jobs. It must note which job need some i/o operation and the status of all jobs being serviced i.e., they are in ready , running or hold state.

- Choosing the policy for transferring the job from hold state to ready state. The decision is made on the basis of priority, resources requested or system balance.
- Allocate the resources for the schedule job by use of memory device and processor management.
- De-allocate the resources when job is completed.

Description of process life cycle Contd…

(7)PROCESS MANAGEMENT CONTD..

JOB SCHEDULING PERFORMANCE

Definition and notation

ⱡ : no of jobs

A : job arrival time

X : execution time required by the job

C : job completion time

(C-A) : job turnaround time (t)

(C-D) : dead line overrun

(C-A)/X : weighted turn around (w)

1/n Σ(C-A) : mean turnaround time ($\overline{t}$)

1/n Σ{(C-A)/x} : mean weighted turn around ($\overline{tw}$)

Throughput = n/turnaround time

n/{max(C)-min(A)}

Operating System

Dr. Satyabrata Das

Associate Professor

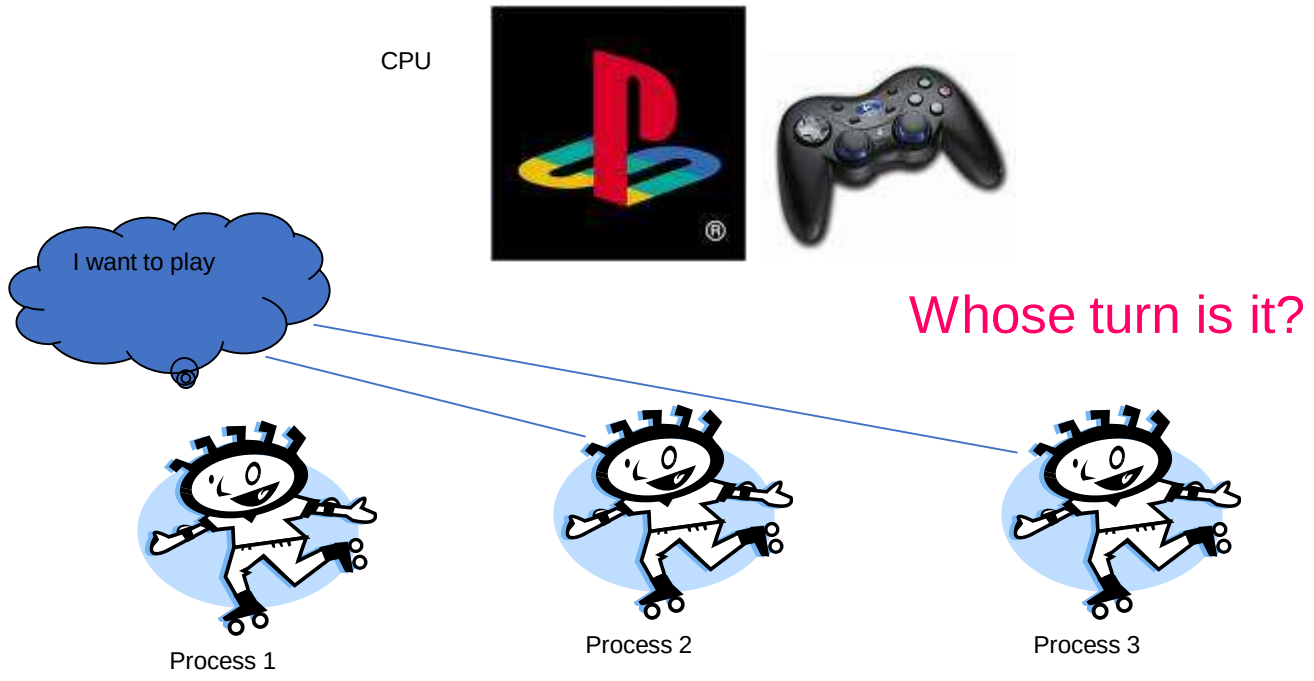
Dept. of IT, VSSUT, Burla

Outlines

- Introduction to CPU Scheduling
- Schedulers
  - (a) Long term Scheduler
  - (b) Short term scheduler
  - (c) Medium term scheduler
- Dispatcher
- Context switching
- Scheduling Criteria/Methodology

Scheduling

- Deciding which process/thread should occupy the resource (CPU, disk, etc)

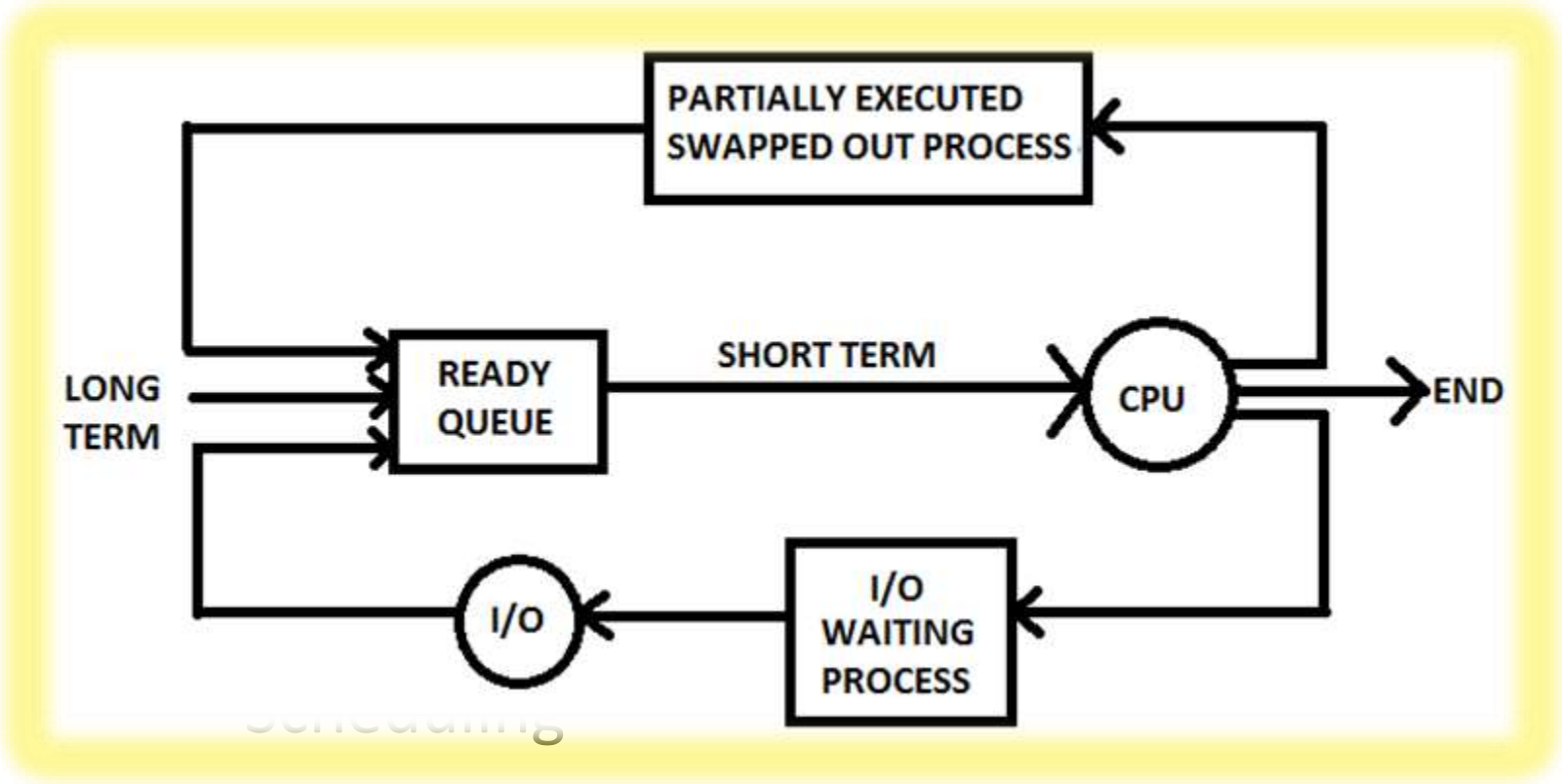


CPU Scheduling

Scheduling is a fundamental function of operating system. When a computer is multiprogrammed it has multiple processes competing for CPU at the same time. If only one CPU is available then a choice has to be made regarding which process has to execute next. This decision making process is known as scheduling and the part of the operating system that makes the choice is known as the scheduler. The algorithm it uses in making this choice is called scheduling algorithm.

CPU scheduling is the basis of multiprogrammed operating system. Scheduling is a fundamental operating system function, almost all computer resources are scheduled before use. The CPU is also one of the primary resources. So CPU is also scheduled before use. The CPU scheduling algorithm determines how the CPU will be allocated to the process.

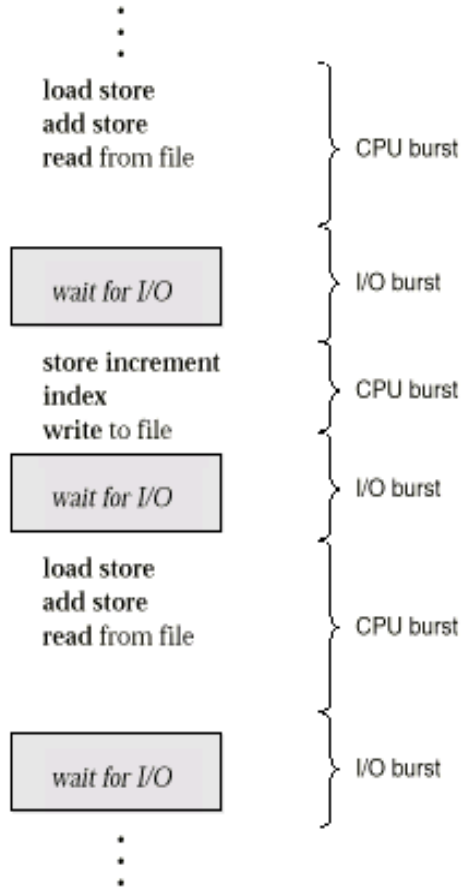
CPU scheduling contd..



CPU SCHEDULING

Scheduling Concepts

Multiprogramming	A number of programs can be in memory at the same time. Allows overlap of CPU and I/O.
Jobs	(batch) are programs that run without user interaction.
User	(time shared) are programs that may have user interaction.
Process	is the common name for both.
CPU - I/O burst cycle	Characterizes process execution, which alternates, between CPU and I/O activity. CPU times are generally much shorter than I/O times.
Preemptive Scheduling	An interrupt causes currently running process to give up the CPU and be replaced by another process.



CPU SCHEDULING

The Scheduler

Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them

- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state
  2. Switches from running to ready state
  3. Switches from waiting to ready
  4. Terminates
- Scheduling under 1 and 4 is *nonpreemptive*
- All other scheduling is *preemptive*

CPU SCHEDULING

The Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program

- *Dispatch latency* – time it takes for the dispatcher to stop one process and start another running

Schedulers

A process migrates between the various scheduling queues throughout its lifetime process is called scheduler. The operating system must select for the scheduling process from these queues in some fashion. This selection process is carried out by the appropriate scheduler. In a batch system more processes are submitted then executed immediately. So these processes are spooled to a mass storage device like disk, where they are kept for later execution.

Types of Schedulers

There are 3 main schedulers

1. Long-term scheduler (jobs scheduler) – selects which programs/processes should be brought into the ready queue.
2. Medium-term scheduler (emergency scheduler) – selects which job/process should be swapped out if system is loaded.
3. Short-term scheduler (CPU scheduler) – selects which process should be executed next and allocates CPU.

Types of Schedulers Contd..

(1) Long Term Scheduler

It selects processes from disk and loads them into memory for execution. It controls the degree of the multiprogramming i.e., no of processes in memory. It executes less frequently than other scheduler. If the degree of multiprogramming is stable than the average rate of process creation is equal to the average departure rate of processes leaving the system. So the long term scheduler is required to be invoked when the process leaves the system. Due to longer interval between executions it can afford to take more time to decide which process should be selected for execution.

Types of Schedulers Contd..

(2)Short Term Scheduler

The short term scheduler selects among the process that are ready to execute and allocate the CPU to one of them. The primary distinction between these two schedulers is the frequency of their execution. The short term scheduler must select a new process for the CPU quite frequently. It must execute at least one in 100ms. Due to the short duration of time between executions, it must be very first.

Types of Schedulers Contd..

(3)Medium Term Scheduler

Some operating systems introduce an additional intermediate level of scheduling known as medium term scheduler. The main idea behind this scheduler is that sometimes it is advantageous to remove process from memory and thus reduce the degree of multiprogramming. At some later time, the process can be reintroduced into memory and its execution can be continued from where it had left off. This is called as swapping. The process is swapped out and swapped in later by medium term scheduler. Swapping is necessary to improve the process miss or due to some change in memory requirements the available memory limit is exceeded which required some memory to be freed up.

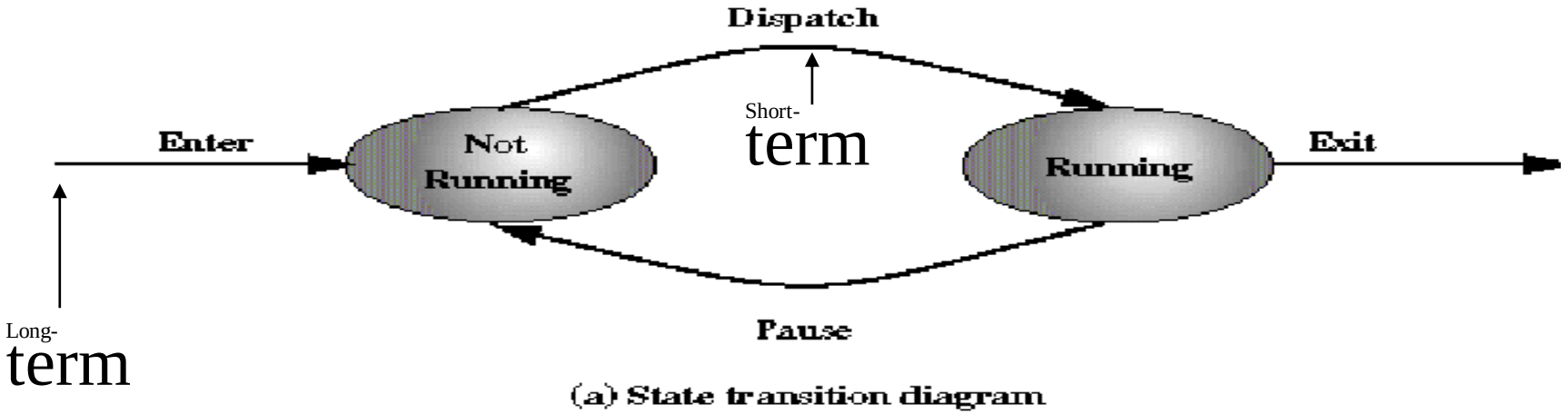
Operating System

Dr. Satyabrata Das

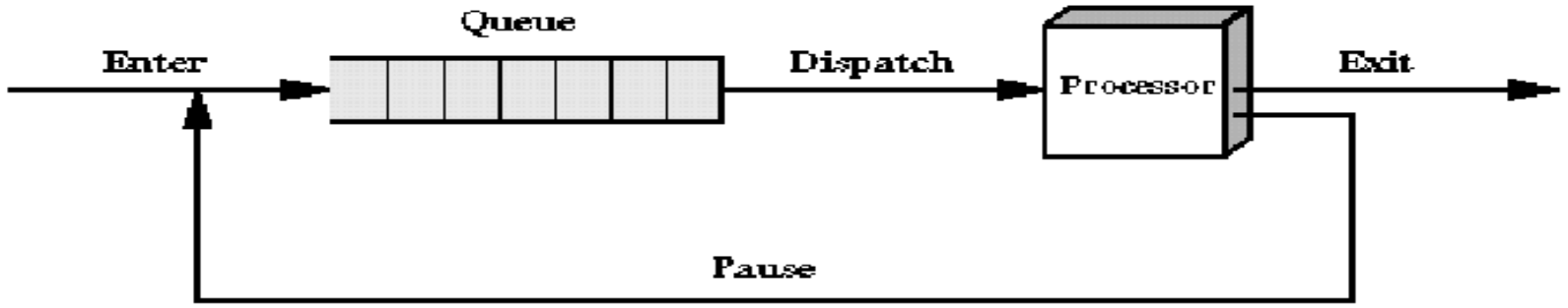
Associate Professor

Dept. of IT, VSSUT, Burla

Long/Short-Term Scheduling



(a) State transition diagram

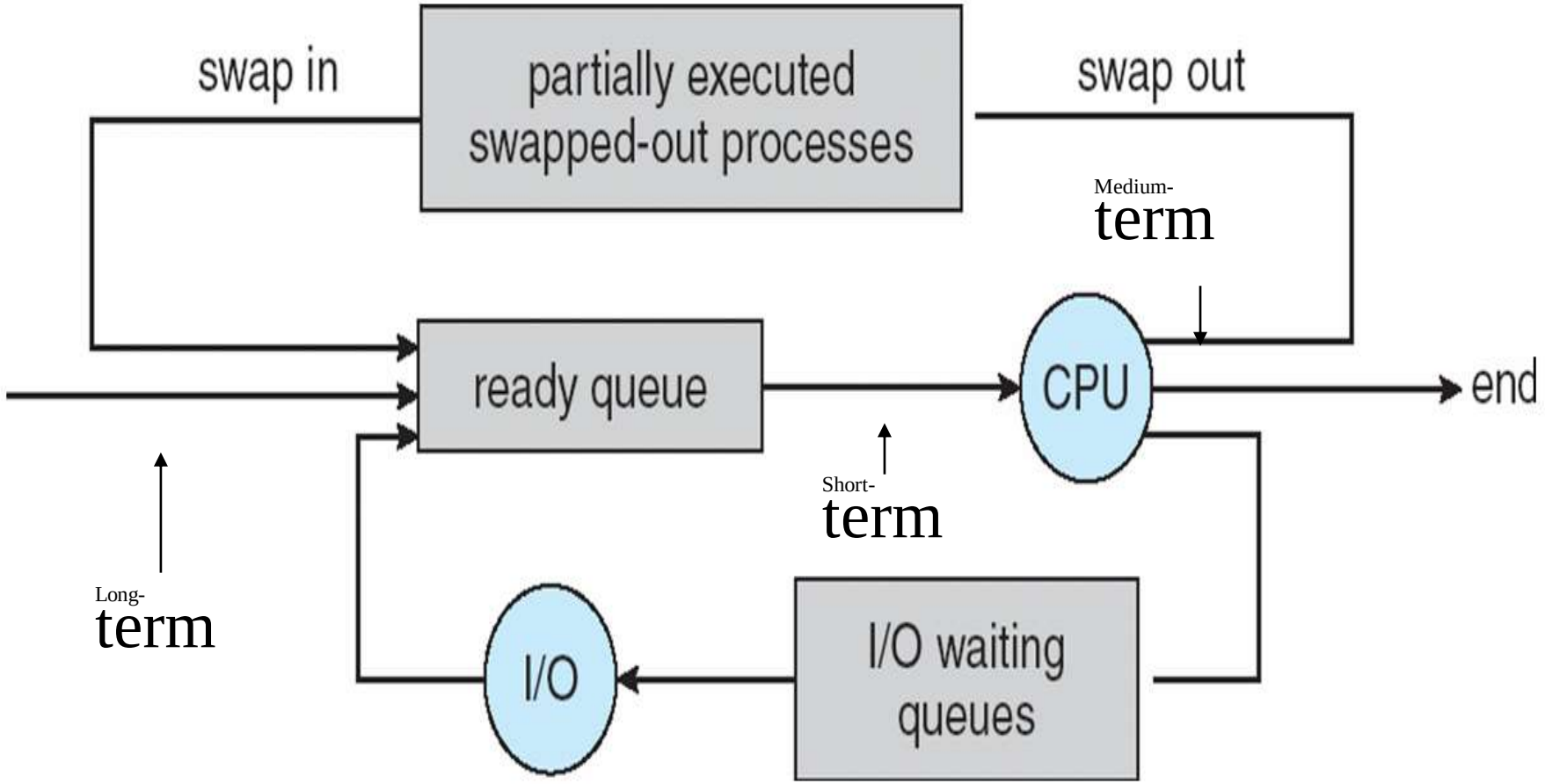


(b) Queuing diagram

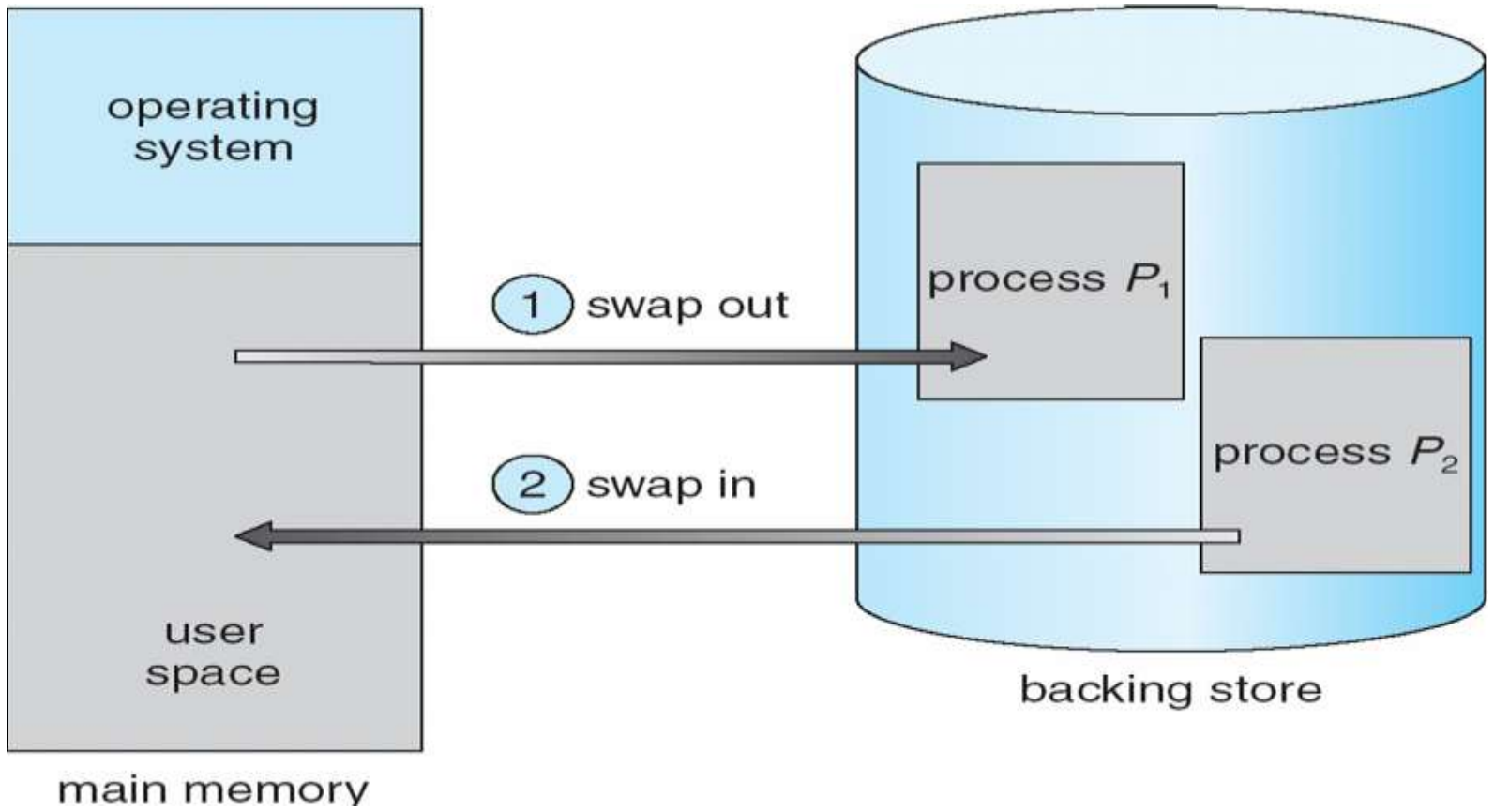
Aspects of Schedulers

- Long-term scheduler is invoked very infrequently (seconds, minutes) $\Rightarrow$ (may be slow).
- The long-term scheduler controls the degree of multiprogramming.
- Short-term scheduler is invoked very frequently (milliseconds) $\Rightarrow$ (must be fast).
- Processes can be described as either:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts.
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts.

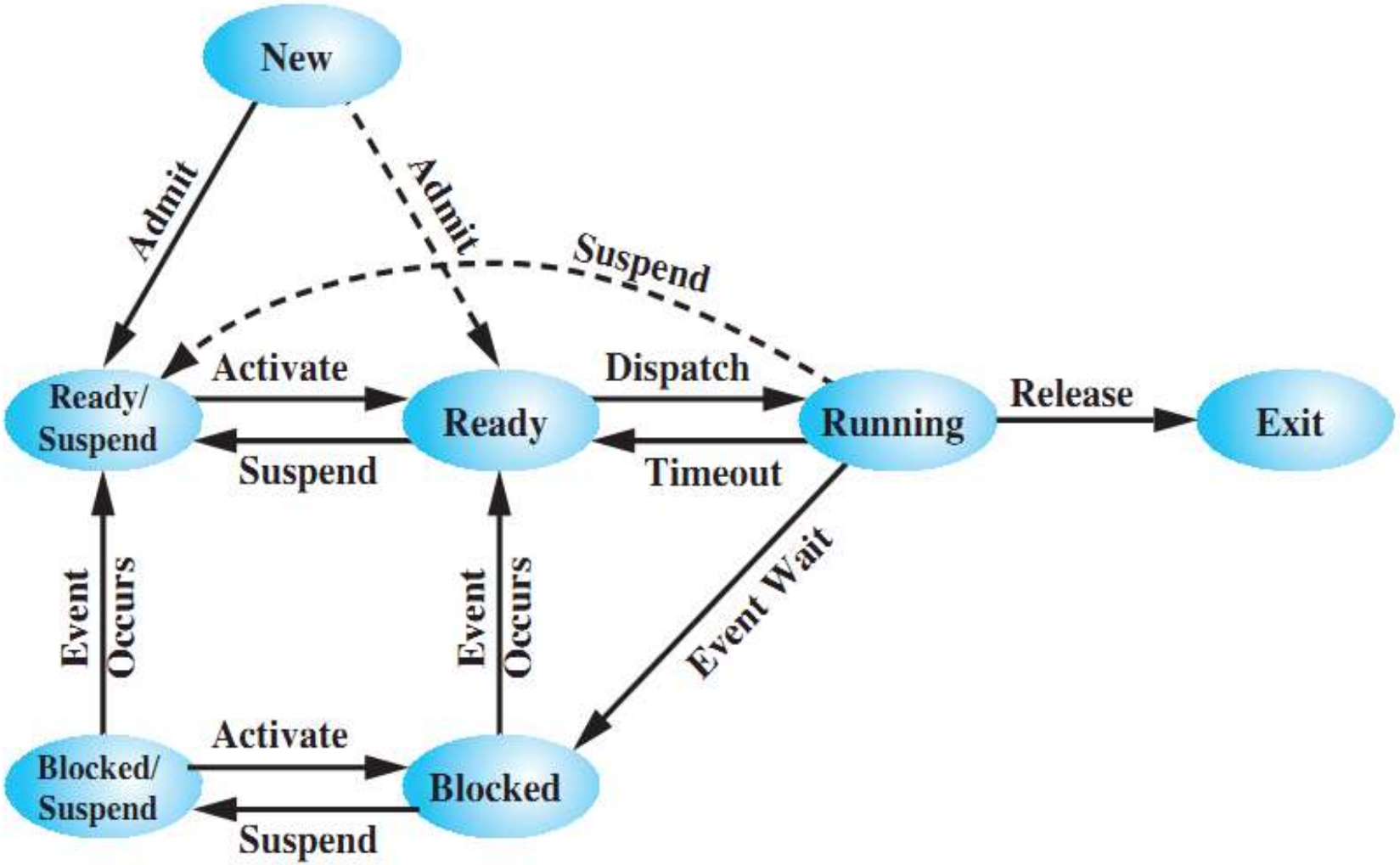
Medium Term Scheduling



Schematic View of Swapping



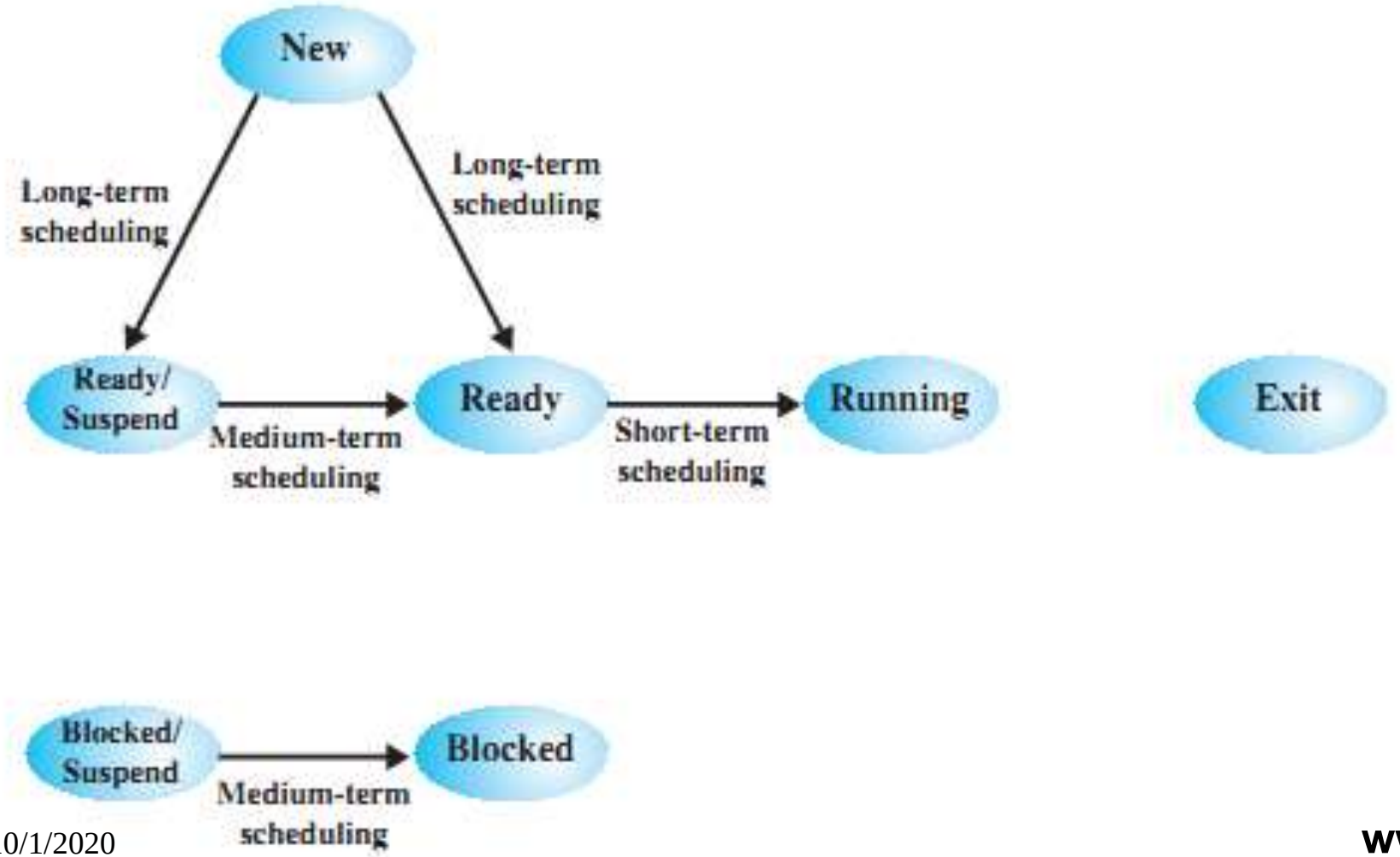
A Seven-state Process Model



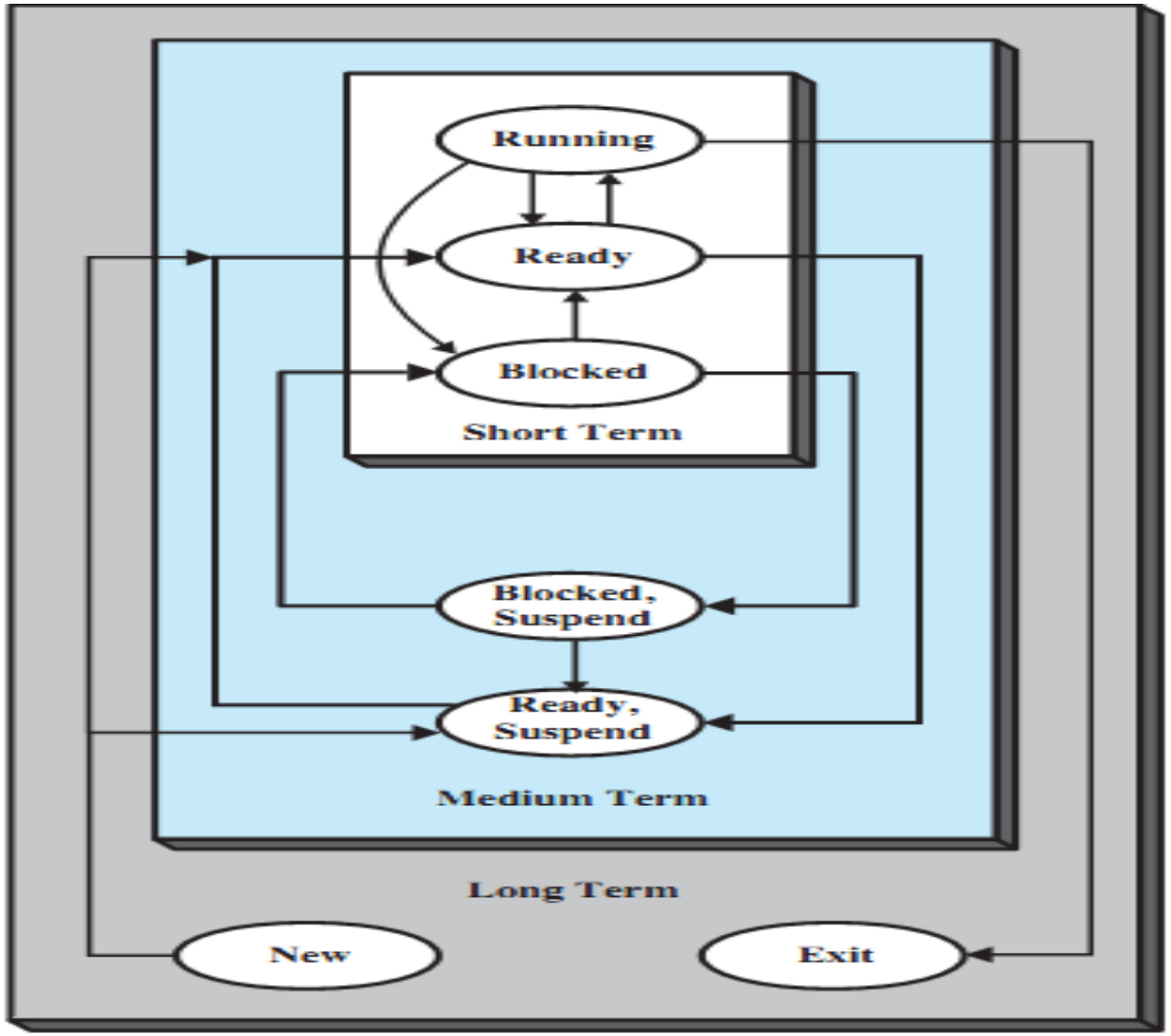
New state transitions

- 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 has been waiting occurs (state info is available to OS).
- Ready Suspend —> Ready
  - when no more ready processes in main memory.
- Ready —> Ready Suspend (unlikely)
  - When there are no blocked processes and must free memory for adequate performance.

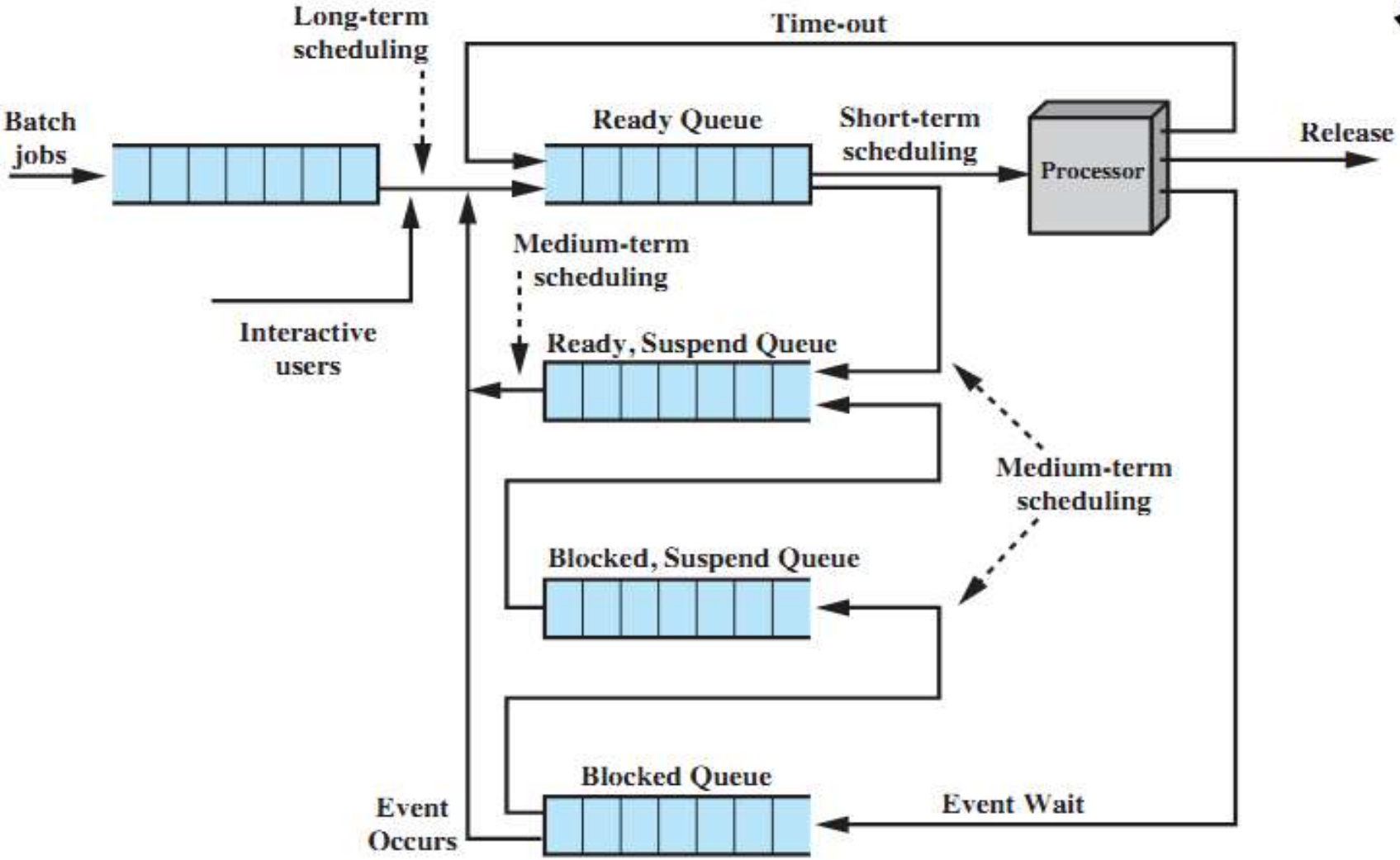
Classification of Scheduling Activity



Another view of the 3 levels of scheduling



Queuing Diagram for Scheduling



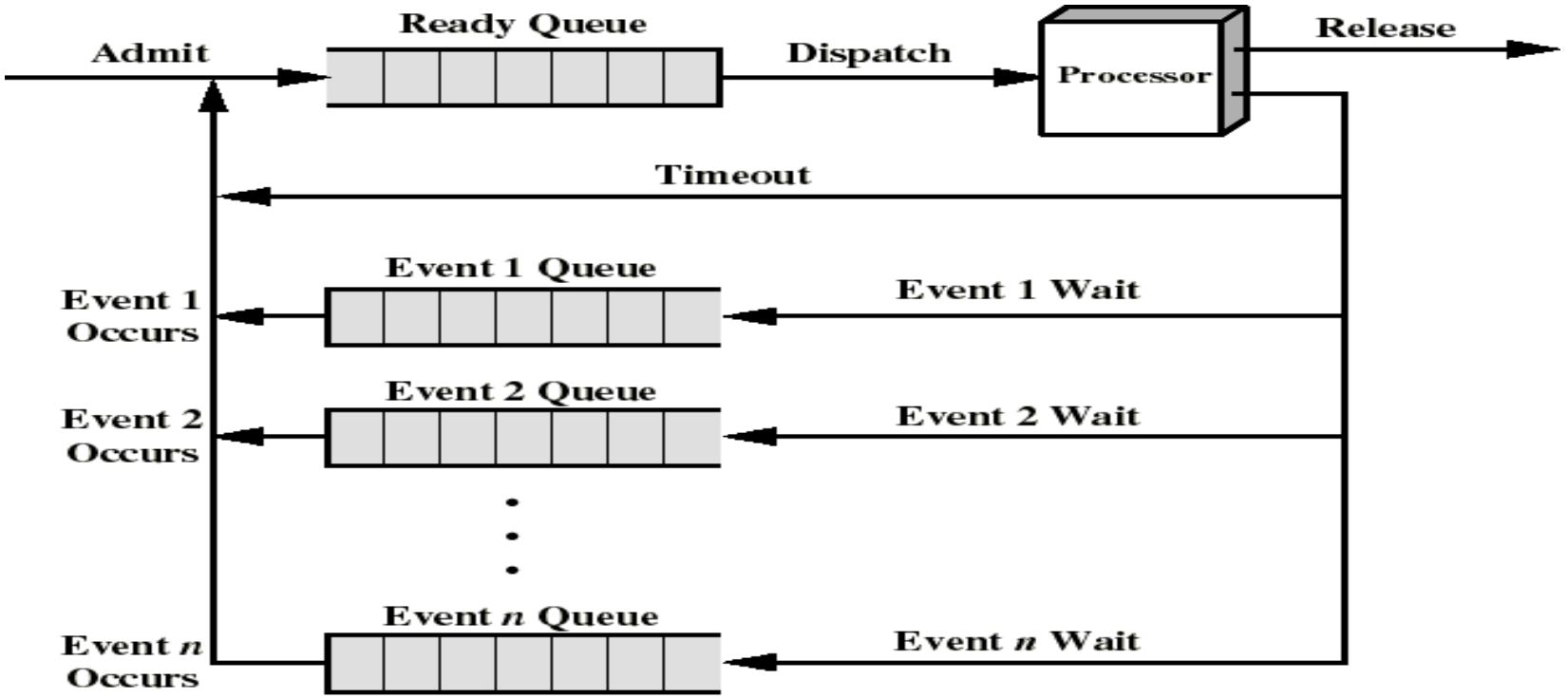
Dispatcher (short-term scheduler)

- 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.
- The CPU will always execute instructions from the dispatcher while switching from process A to process B.

Process Scheduling Queues

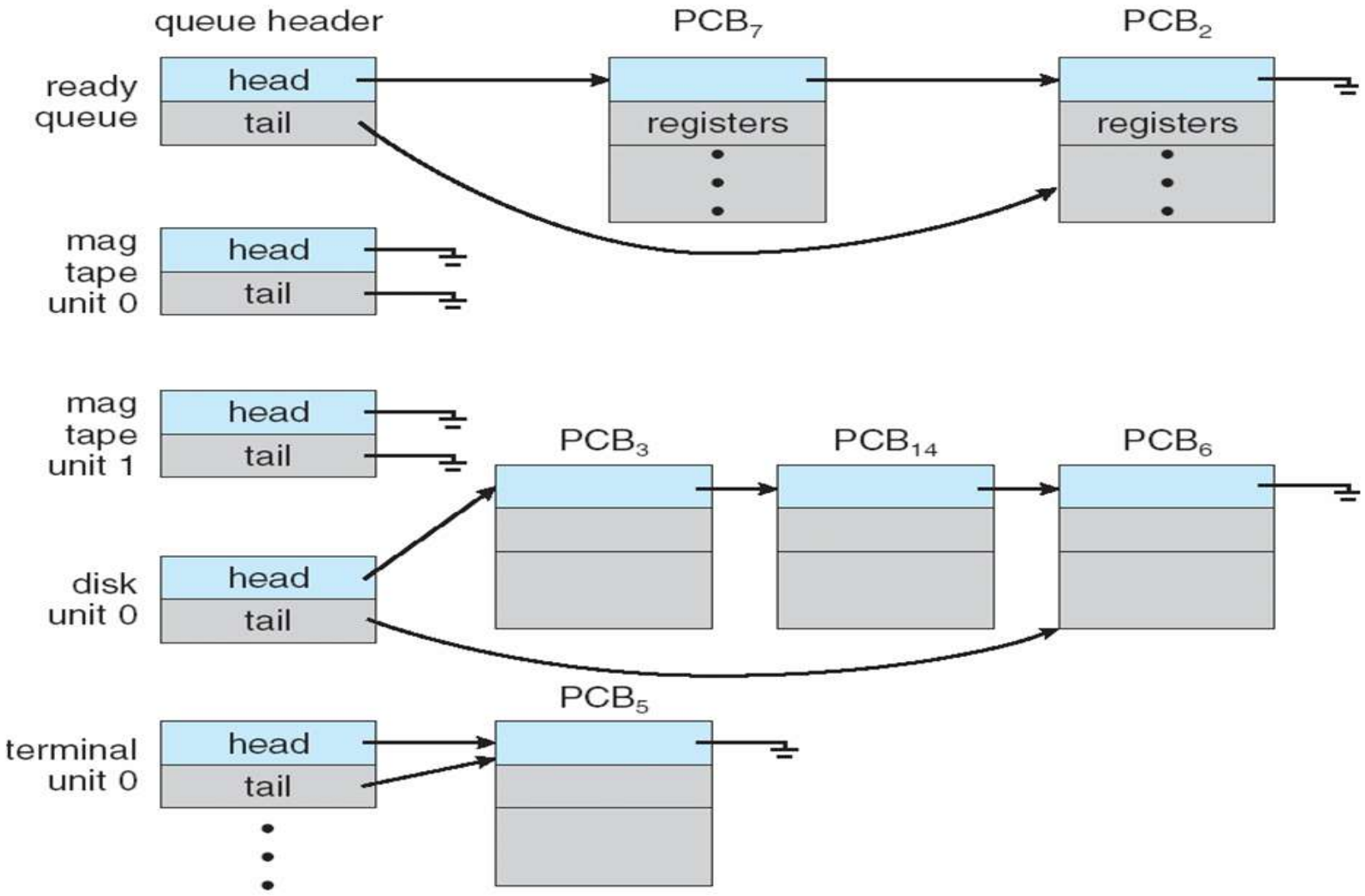
- Process queue – set of *all* processes in the system.
- Ready queue – set of processes residing in main memory, ready and waiting to execute.
- Device queues – set of processes waiting for an I/O device.
- Processes migrate among the various queues.

A Queuing Discipline



- **When event n occurs, the corresponding process is moved into the ready queue**

Ready Queue and various I/O Device Queues



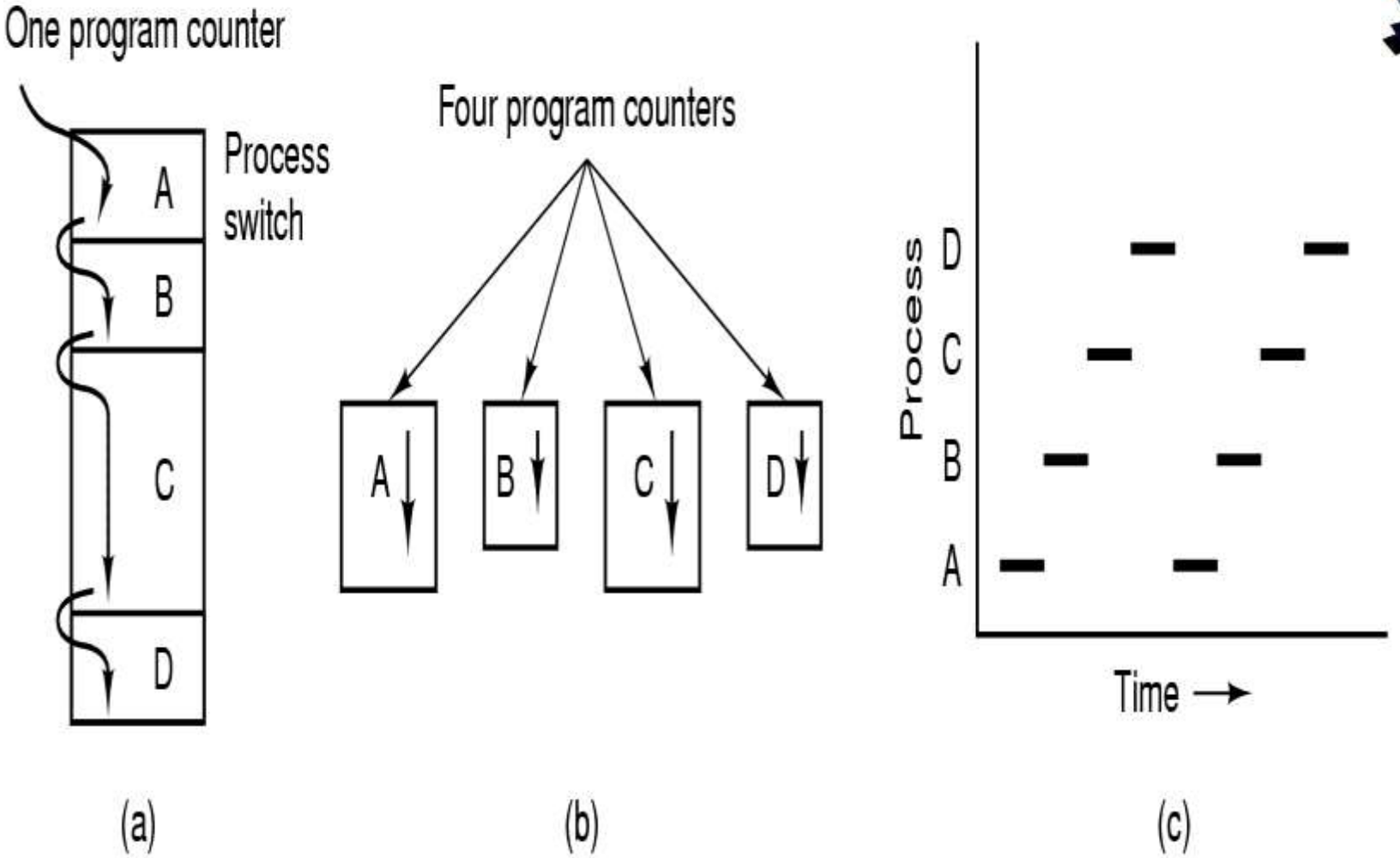
Operating System

Dr. Satyabrata Das

Associate Professor

Dept. of IT, VSSUT, Burla

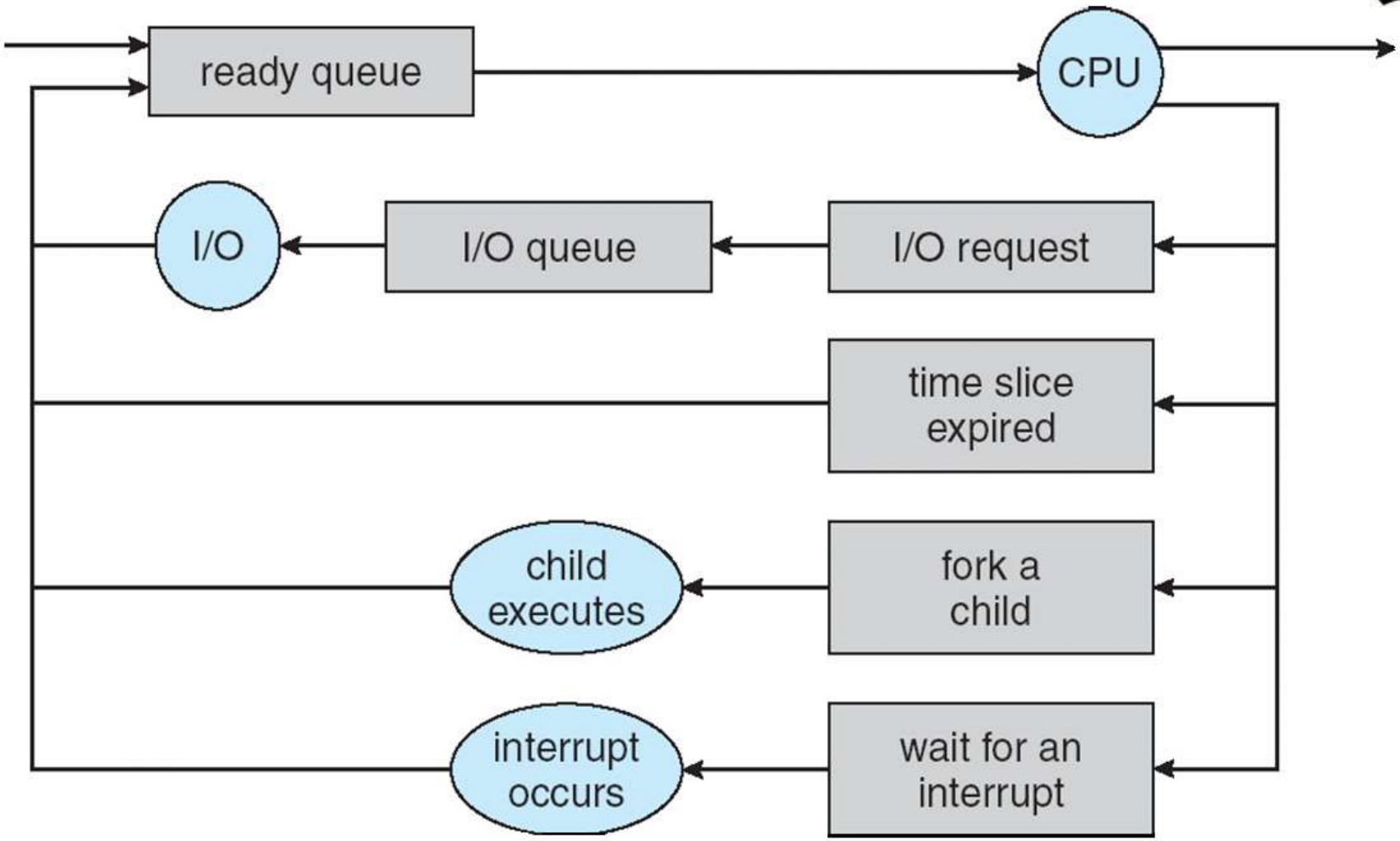
Process Switch



When to Switch a Process?

- A process switch may occur whenever the OS has gained control of CPU. i.e., when:
  - Supervisor Call
    - explicit request by the program (example: 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 terminated state.
  - Interrupt
    - the cause is external to the execution of the current instruction – control is transferred to Interrupt Handler.

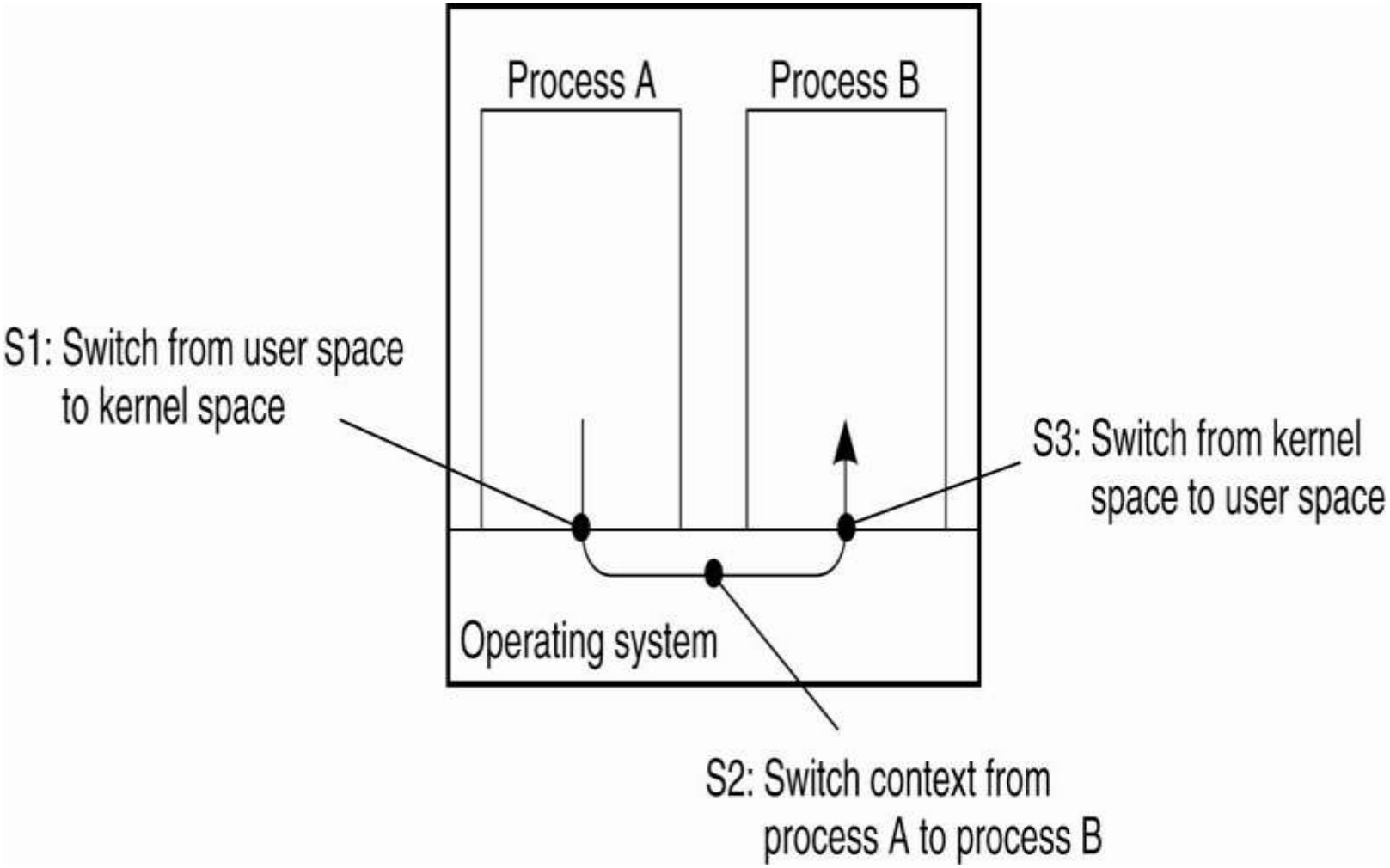
Reasons for Process Switch



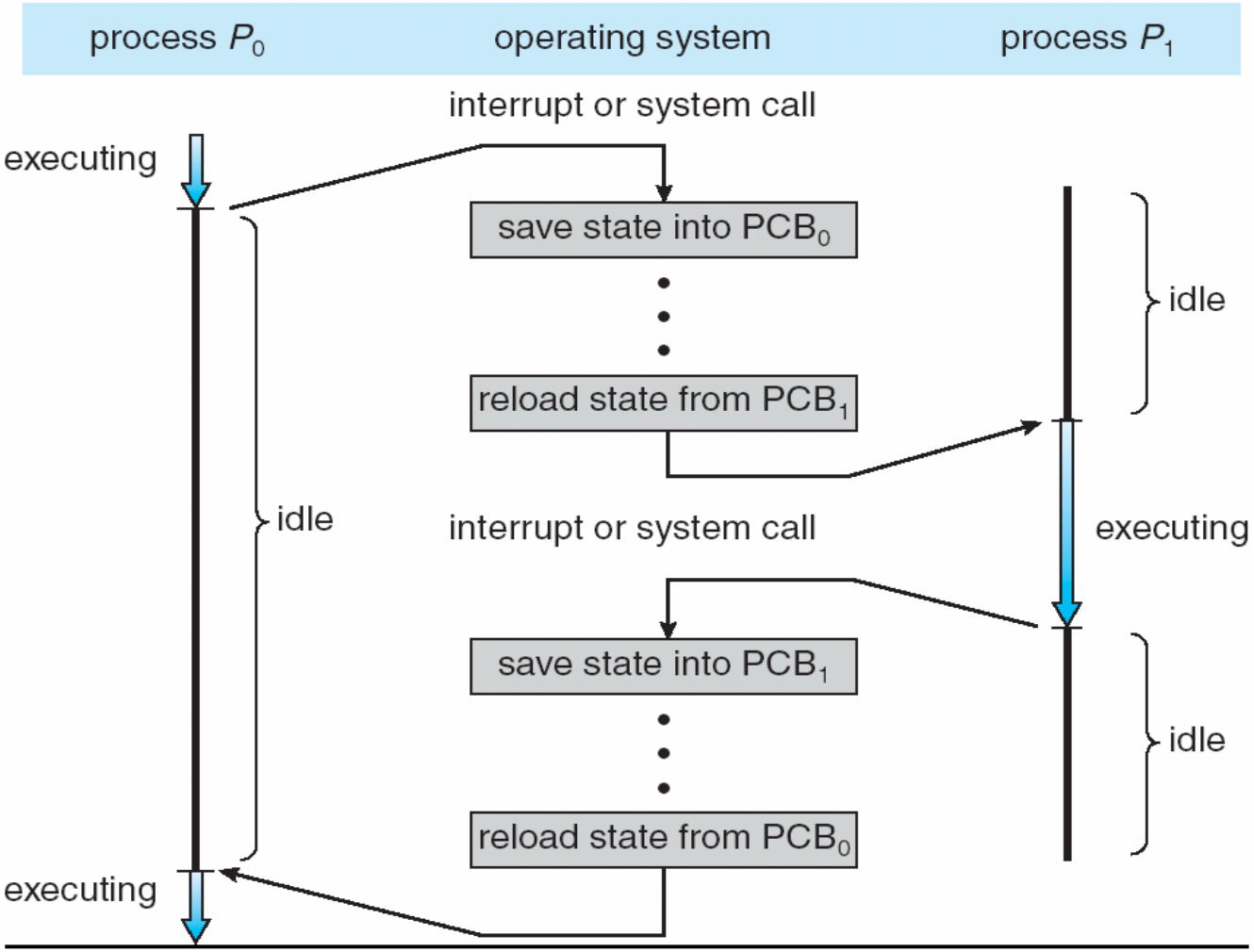
Context Switch

- When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process.
- This is called context switch.
- Context of a process represented in the PCB.
- The time it takes is dependent on hardware support.
- Context-switch time is overhead; the system does no useful work while switching.

Context switch between processes (1)



Context switch between processes (2)



Steps in Context Switch

- Save context of processor including program counter and other registers.
- Update the PCB of the running process with its new state and other associate information.
- 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.

Scheduling Criteria/ Methodology

- CPU Utilization
  - Keep the CPU and other resources as busy as possible
- Throughput
  - # of processes that complete their execution per time unit.
- Turnaround time
  - amount of time to execute a particular process from its entry time.
  - The time interval between the submission of a process and the time of completion is the turnaround time.

  TAT=Waiting time in ready queue + executing time + Waiting time in waiting queue for I/O

Scheduling Criteria (cont.)

- Waiting time
  - amount of time a process has been waiting in the ready queue.
- Response Time (in a time-sharing environment)
  - amount of time it takes from when a request was submitted until the first response is produced, NOT output.

Optimization Criteria

- Optimize overall system
  - Max CPU Utilization
  - Max Throughput
- Optimize individual processes' performance
  - Min Turnaround time
  - Min Waiting time
  - Min Response time

Operating System

Dr. Satyabrata Das

Associate Professor

Dept. of IT, VSSUT, Burla

Outlines

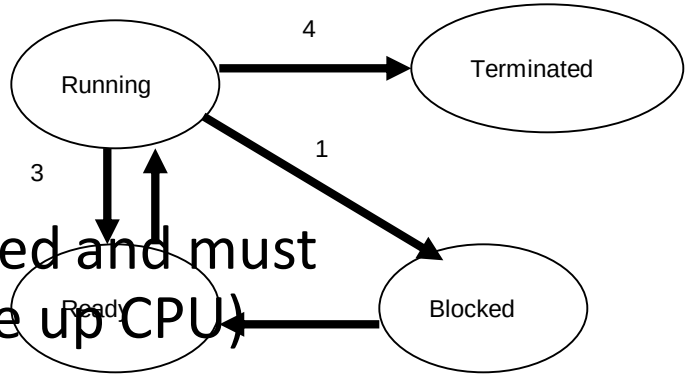
- Introduction to CPU Scheduling
- Preemptive vs Non-Preemptive scheduling
- First come First service(FCFS) scheduling
- Shortest job first(SJF) scheduling
- Shortest remaining time first(SRTF) scheduling
- Round Robin(RR) scheduling
- Priority Scheduling
- Multilevel queue scheduling
- Multilevel feedback queue scheduling
- Multi Processor scheduling
- Real-time scheduling

CPU Scheduling Algorithms

CPU scheduling algorithms decides which of the process in the ready queue is to be allocated the CPU. There are many different CPU scheduling algorithms, out of those algorithms, which algorithm maximize the CPU utilization and throughput and minimize turnaround time, waiting time and response time, those algorithms are the best of all algorithms

Preemptive vs. Non-preemptive scheduling

- **Non-preemptive scheduling:**
  - The running process keeps the CPU until it voluntarily gives up the CPU
    - process exits
    - switches to blocked state
    - Transition 3 is only voluntary
- **Preemptive scheduling**:
  - The running process can be interrupted and must release the CPU (can be forced to give up CPU)



Preemptive vs. Non-preemptive scheduling

- **Non-preemptive scheduling:**

  In non preemptive scheduling a scheduled job always completes before another scheduling decision is made. The jobs therefore finish in the order in which they are scheduled. The example of non preemptive scheduling are first come first serve (FCFS), shortest job first (SJF), priority algorithms.

  In non-preemptive once the CPU assigned to a process, the processor do not release until the completion of that process. The CPU will assigned to some other job only after the previous job has finished

- **Preemptive scheduling**:

  In preemptive scheduling the CPU can release the processes even in the middle of execution

CPU Scheduling Algorithms

- First-Come First Served scheduling(FCFC)
- Shortest job first scheduling(SJF)
- Shortest remaining time first(SRTF)
- Round Robin Scheduling(RR)
- Priority Scheduling
- Multilevel Queue Scheduling
- Multilevel Feedback Queue Scheduling

First Come First Serve (FCFS) Scheduling

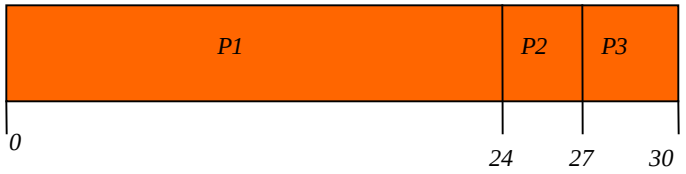
- Policy: Process that requests the CPU FIRST is allocated the CPU FIRST.
  - FCFS is a non-preemptive scheduling algorithm.
- Implementation - using FIFO queues
  - incoming process is added to the tail of the queue.
  - Process selected for execution is taken from head of queue.
- Performance metric - Average waiting time in queue.
- Gantt Charts are used to visualize schedules.

First-Come, First-Served(FCFS) Scheduling

- ***Example***

Process	Burst Time
P1	24
P2	3
P3	3

Gantt Chart for Schedule



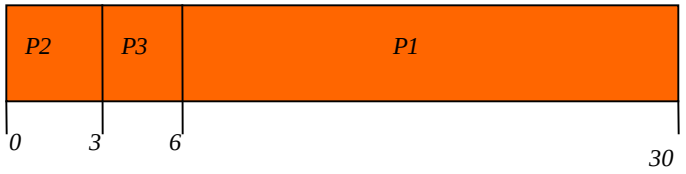
- Suppose the arrival order for the processes is
  - P1, P2, P3
- Waiting time
  - P1 = 0;
  - P2 = 24;
  - P3 = 27;
- Average waiting time
  - (0+24+27)/3 = 17

FCFS Scheduling (cont.)

- ***Example***

Process	Burst Time
P1	24
P2	3
P3	3

Gantt Chart for Schedule



- Suppose the arrival order for the processes is
  - P2, P3, P1
- Waiting time
  - P1 = 6; P2 = 0; P3 = 3;
- Average waiting time
  - (6+0+3)/3 = 3 , better..
- Convoy Effect:
  - short process behind long process, e.g. 1 CPU bound process, many I/O bound processes.

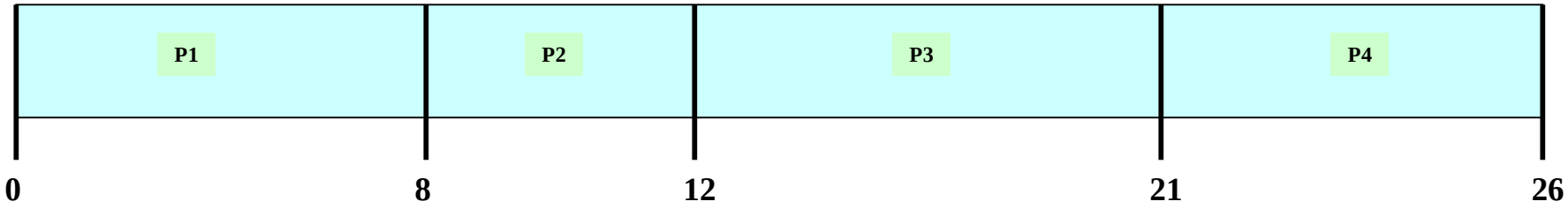
CPU SCHEDULING

Scheduling Algorithms

EXAMPLE DATA:

Process	Arrival Time	Service Time
1	0	8
2	1	4
3	2	9
4	3	5

FCFS



Average wait = ((8-0) + (12-1) + (21-2) + (26-3))/4 = 61/4 = 15.25

Residence Time at the CPU