- ECE 405 MICROCOMPUTER & MICROPROCESSOR

WEEK FOUR

CPU Scheduling

- CPU scheduling is the basis of multiprogrammed operating systems. By switching the CPU among processes, the operating system can make the computer more productive.
- Scheduling is Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them

Scheduling Objectives

- Enforcement of fairness in allocating resources to processes
- Enforcement of priorities
- Make best use of available system resources
- Give preference to processes holding key resources.
- Give preference to processes exhibiting good behavior.
- Degrade gracefully under heavy loads

TYPES OF SCHEDULING

- Non-preemptive Scheduling

  Once CPU has been allocated to a process, the process keeps the CPU until Process exits OR

  Process switches to waiting state
- Preemptive Scheduling

  Process can be interrupted and must release the CPU.

  Need to coordinate access to shared data

CPU Scheduling Decisions

CPU scheduling decisions may take place when a process:

1. switches from running state to waiting state (for example, I/O request, or invocation of wait for the termination of one of the child processes).
2. switches from running state to ready state (for example, when an interrupt occurs).
3. switches from waiting to ready (for example, completion of I/O).

CPU Scheduling Decisions

4. When a process terminates.

- Scheduling under 1 and 4 is non-preemptive.

All other scheduling is preemptive

CPU-I/O Burst Cycle

- Maximum CPU utilization obtained with multiprogramming.
- Process execution consists of a cycle of CPU execution and a cycle of I/O wait.

The process involved in this switch includes

- **Context Switch**
- **Dispatcher**

CPU-I/O Burst Cycle Process

Context Switch

- To give each process on a multiprogrammed machine a fair share of the CPU, a hardware clock generates interrupts periodically.
- This allows the operating system to schedule all processes in main memory (using scheduling algorithm) to run on the CPU at equal intervals. Each switch of the CPU from one process to another is called a context switch

CPU-I/O Burst Cycle Process

Dispatcher

Dispatcher module gives control of the CPU to the process selected by the short-term scheduler. This involves:

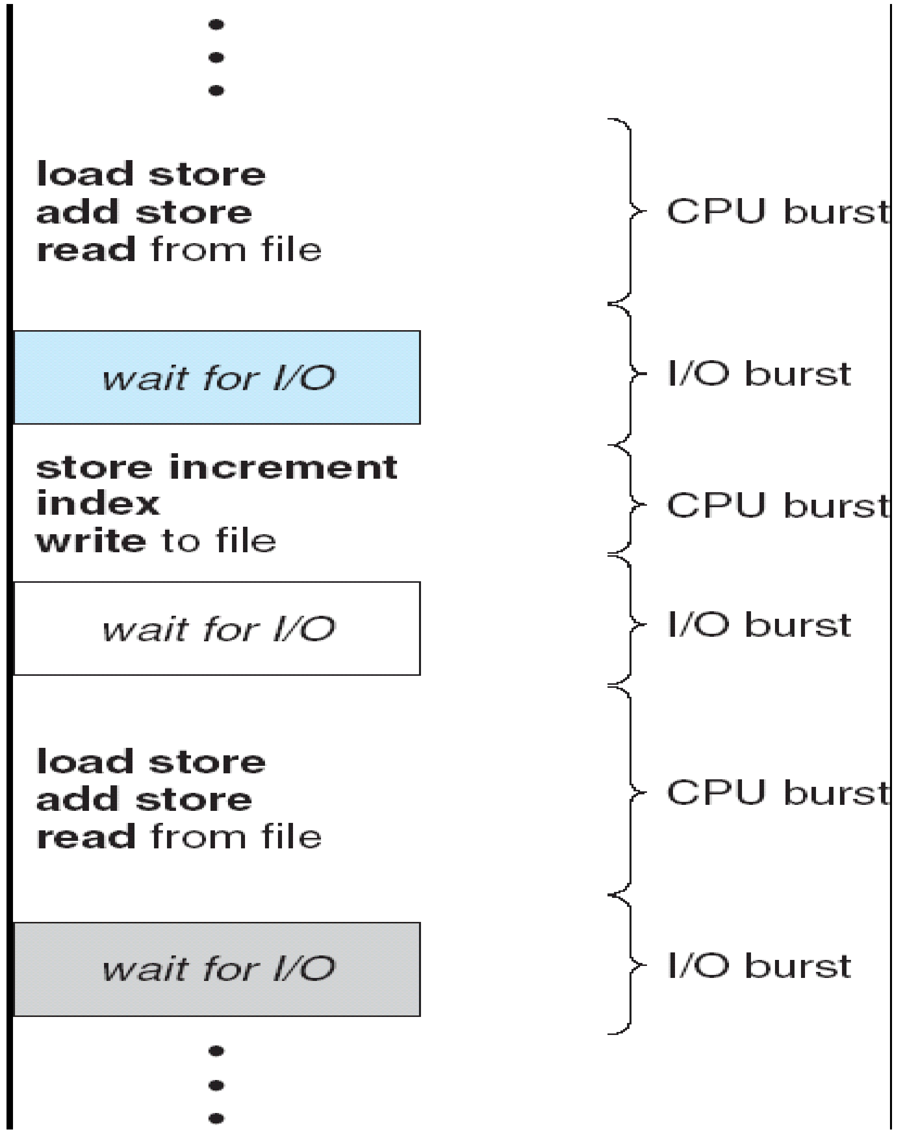
- *Switching context.*
- *Switching to user mode.*
- *Jumping to the proper location in the user program to restart that program*

CPU-I/O Burst Cycle Process

Dispatch Latency:

- This is the time it takes for the dispatcher to stop one process and start another running.
- Dispatcher must be fast

CPU-I/O Burst Cycle



Levels of Scheduling

- High Level Scheduling or Job Scheduling

  Selects jobs allowed to compete for CPU and other system resources.

- Intermediate Level Scheduling or Medium Term Scheduling

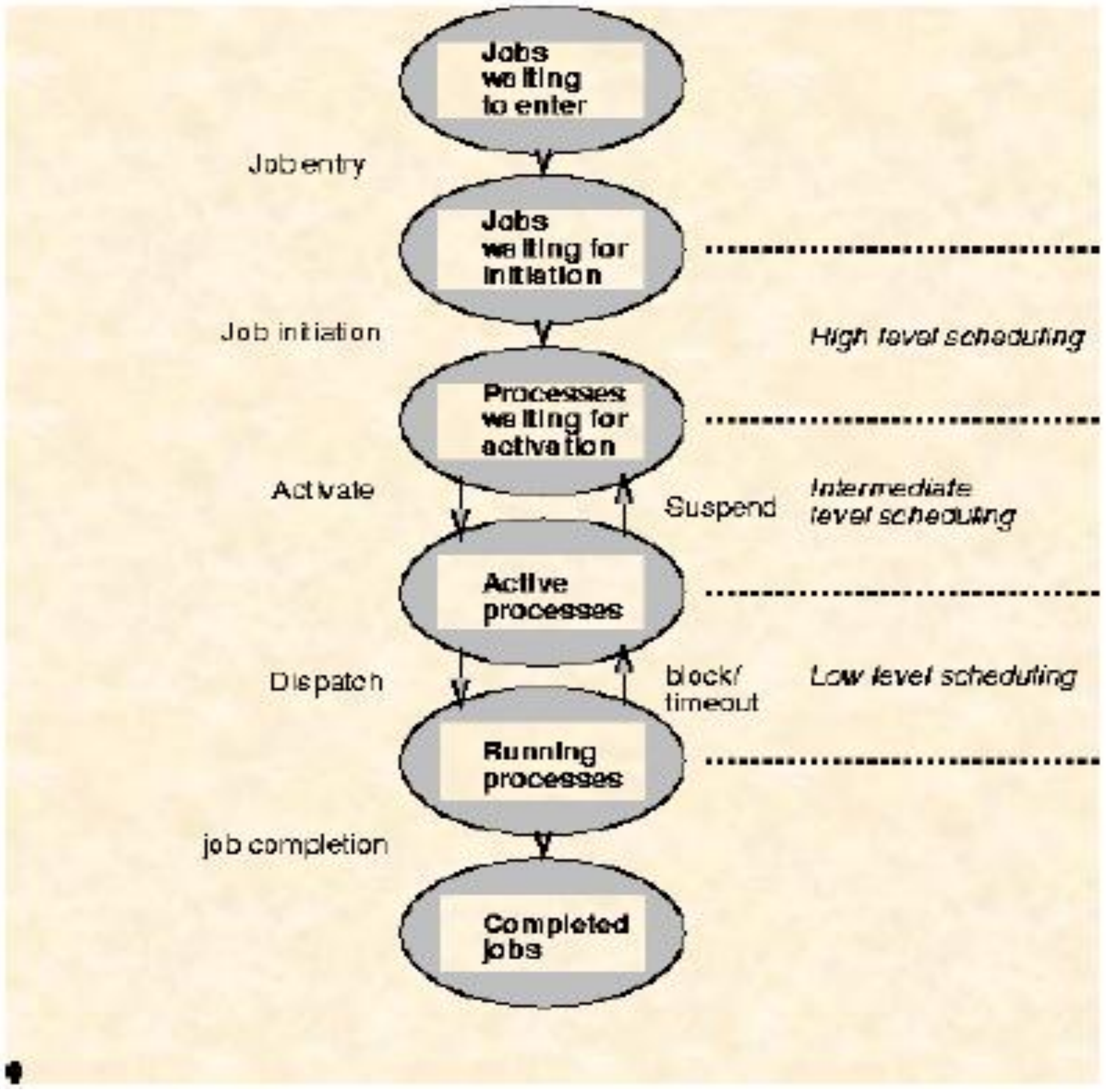
  Selects which jobs to temporarily suspend/resume to smooth fluctuations in system load

Levels of Scheduling

- Low Level (CPU) Scheduling or Dispatching

  Selects the ready process that will be assigned the CPU.

Levels of Scheduling (cont')



Scheduling Criteria

Different CPU scheduling algorithms have different properties and may favor one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms. Criteria that are used include the following

1. CPU utilization.

- Keep the CPU and other resources as busy as possible

Scheduling Criteria (cont')

2. ***Throughput.*** *N*umber of processes that complete their execution per time unit
3. ***Turnaround time.*** Amount of time to execute a particular process from its entry time.
4. ***Waiting time.*** Amount of time a process has been waiting in the ready queue
5. ***Response time.*** Amount of time it takes from when a request was submitted until the first response is produced, NOT output

Scheduling Criteria (cont’)

To Optimization Criteria

- Maximize CPU Utilization
- Maximize Throughput
- Minimize Turnaround time
- Minimize Waiting time
- Minimize response time

Scheduling Algorithms

1. *First-Come, First-Served Scheduling (FCFS)*
2. *Shortest-Job-First Scheduling (SJF)*
3. *Priority Scheduling*
4. *Round-Robin Scheduling*
5. *Multilevel Queue Scheduling*
6. *Multilevel Feedback Queue Scheduling*

Scheduling Algorithms (cont')

(FCFS) Scheduling Policy: Process that requests the CPU *FIRST is allocated the CPU FIRST.*

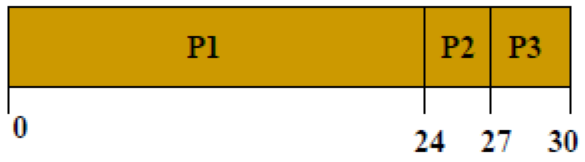
- FCFS is a non-preemptive algorithm.
- Implementation - using FIFO queues incoming process is added to the tail of the queue.
- Process selected for execution is taken from head of queue.
- Performance metric - Average waiting time in queue. Gantt Charts are used to visualize schedules

Scheduling Algorithms (cont')

- 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

Average completion time

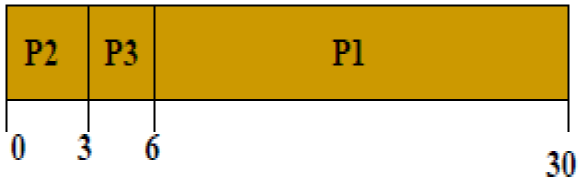
- (24+27+30)/3 = 27

Scheduling Algorithms (cont')

Example 2

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..

Average waiting time

- (3+6+30)/3 = 13 , better..

Convoy Effect:

- short process behind long process, e.g. 1 CPU bound process, many I/O bound processes.

Scheduling Algorithms (cont')

- I/O boundedness

short burst of CPU before blocking for I/O

- CPU boundedness

extensive use of CPU before blocking for I/O

Scheduling Algorithms (cont')

Round Robin (RR)

- Each process gets a small unit of CPU time
- Time quantum usually 10-100 milliseconds.
- After this time has elapsed, the process is preempted and added to the end of the ready queue.
- *n processes, time quantum = q*
- Each process gets 1/*n CPU time in chunks of at most q time units at a time.*

Scheduling Algorithms (cont')

Round Robin (RR)

- No process waits more than (*n-1)q time units.*
- Performance
- Time slice *q too large – response time poor*
- Time slice (∞*)? -- reduces to FIFO behavior*
- Time slice *q too small - Overhead of context switch is too expensive. Throughput poor*
- RR is good for short jobs but cause delays for long jobs

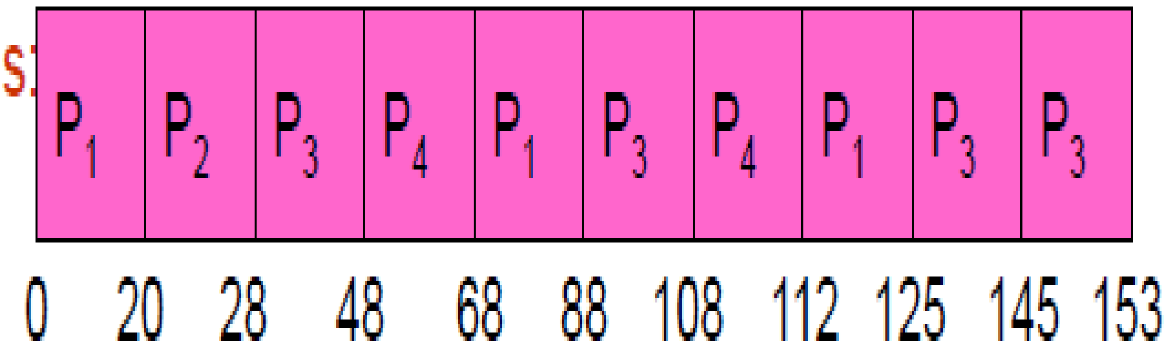
Scheduling Algorithms (cont')

Example of RR with Time Quantum = 20

Example:

Process	Burst Time
P_1	53
P_2	8
P_3	68
P_4	24

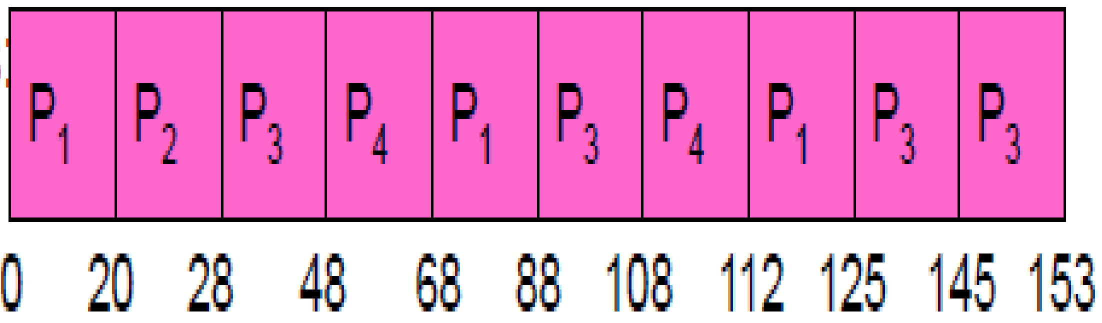
- The Gantt chart is:



- Waiting time

Scheduling Algorithms (cont')

- The Gantt chart is:



- Waiting time
  - P_1=(68-20)+(112-88)=72
  - P_2=(20-0)=20
  - P_3=(28-0)+(88-48)+(125-108)=85
  - P_4=(48-0)+(108-68)=88
- Average waiting time = (72+20+85+88)/4=66¼
- Average completion time = (125+28+153+112)/4 = 104½

Scheduling Algorithms (cont')

Shortest-Job-First(SJF) Scheduling;Associate with each process the length of its next CPU burst.

- Use these lengths to schedule the process with the shortest time.
- SJF operates in two Schemes:

Scheme 1: Non-preemptive

- Once CPU is given to the process it cannot be preempted until it completes its CPU burst.

Scheduling Algorithms (cont')

- Scheme 2: Preemptive

If a new CPU process arrives with CPU burst length less than remaining time of current executing process, preempt. ***Also called Shortest-Remaining-Time-First (SRTF)..***

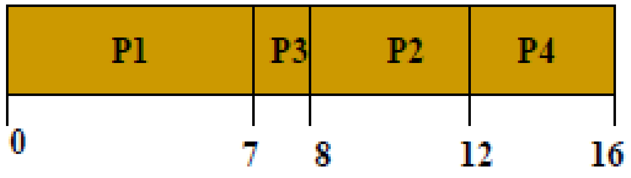
Scheduling Algorithms (cont')

SJF and SRTF (Example)

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

Non-Preemptive SJF Scheduling

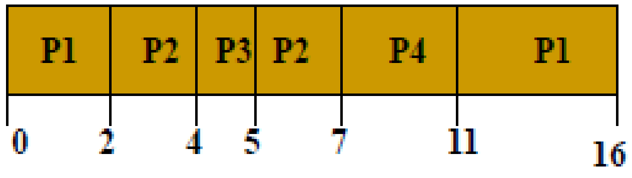
Gantt Chart for Schedule



Average waiting time = (0+6+3+7)/4 = 4

Preemptive SJF Scheduling

Gantt Chart for Schedule



Average waiting time = (9+1+0+2)/4 = 3

Scheduling Algorithms (cont')

Priority Scheduling

- A priority value (integer) is associated with each process. Can be based on
    - Cost to user
    - Importance to user
    - Aging
    - %CPU time used in last X hours.
- CPU is allocated to process with the highest priority.
    - Preemptive
    - Nonpreemptive

Scheduling Algorithms (cont')