ELEC 377 – Operating System

Week 4 – Class 2

Last Class

- Monitors
- Java Synchronization
- Introduction to Scheduling

Today

- Scheduling

Scheduling – Basic Concepts

- Goal: Maximum CPU utilization
  - ◊ give CPU to another process while other is waiting I/O

- Processes proceed in bursts
  - ◊ Do some work
  - ◊ Do some I/O
  - ◊ repeat

Dispatcher

- *Dispatcher* is the part of the scheduler responsible for performing the context switch and resuming the process

- *Dispatch Latency*

◊ time for dispatcher to run

Scheduling Criteria

- CPU utilization – keep CPU as busy as possible
- Throughput – # of jobs done per time unit
- Turnaround Time – Time of submission to Time of Completion
- Waiting Time – amount of time in ready queue
- Response Time – submit time to time of first output request

Estimating CPU Burst Times

- Use length of last CPU burst – exponential average

t_n = current CPU burst time

τ_0 = initial estimate

τ_n = predicted for current burst

τ_{n+1} = prediction for next CPU burst

α = weighting parameter

$$\tau_{n+1} = \alpha \, t_n + (1 - \alpha) \, \tau_n$$

$\alpha = 0 \Rightarrow \tau_{n+1} = \tau_0$ (initial estimate) never changes

$\alpha = 1 \Rightarrow \tau_{n+1} = t_n$ (last time slice) only used

Estimating CPU Burst Times

- Use length of last CPU burst – exponential average

$|_0 = 10$

$t_0 = 5, t_1 = 5, t_2 = 5, t_3 = 8, t_4 = 8$

$\langle = 0.3$

$|_1 = 0.3 * 5 + (0.7) * 10 = 8.5$

$|_2 = 0.3 * 5 + (0.7) * 8.5 = 7.45$

$|_3 = 0.3 * 5 + (0.7) * 7.45 = 6.715$

$|_2 = 0.3 * 8 + (0.7) * 6.715 = 7.1005$

$|_2 = 0.3 * 8 + (0.7) * 7.1005 = 7.37035$

Estimating CPU Burst Times

- predicted time always lags real time
- If process spends a reasonable period of time at a constant burst range then estimate approaches current burst time
- what is reasonable? how to tune?

◊ 〈 is the tuning parameter

◊ 〈 is low, then past behaviour has heavier weight, estimate is slower to change
- ignore transient behaviour

◊ 〈 is high, then last time slice has heavier weight, estimate is faster to change
- faster to adapt to changes

Round Robin (RR)

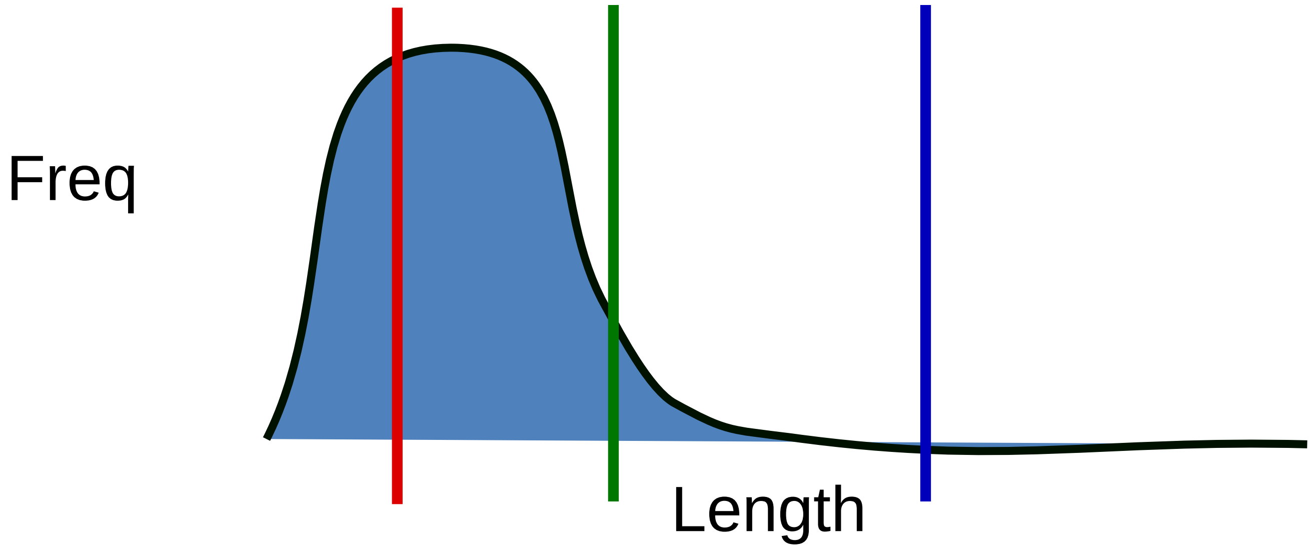
- Similar to FCFS, but add preemption.
- Designed for Time Sharing Systems
- Time slice (*quantum*) maximum time process gets to run
- if the quantum (q) is large FCFS
- if q is small, then appears to be multiple slower CPU's (processor sharing).
- context switching is not free

◊ shorter q, more context switches to complete a single CPU burst for a given process

◊ q must be large with respect to context switch time

◊ 80% of CPU bursts should be shorter than q.

Round Robin – Quantum Length



multilevel Queues

highest priority

- System Processes
- Interactive Processes
- Interactive Editing Processes
- Batch Processes
- Experimental Processes

lowest priority

Multi Level Queues

- Each queue has it’s own scheduling algorithm
- Interactive (foreground) - Round Robin
- Scheduling must be done between the queues
- ◊ usually fixed priority preemptive scheduling (starvation)
- ◊ time slice between queues (portion time between queues)
- In simplest form, processes are assigned a queue and remain there until completion
- Higher priority queues may require more money, or more status

Multi Level Feedback

- processes move between queues
- when doing I/O, processes move to higher priority queues
- When CPU intensive, processes move to lower priority queues
- Give higher priority queues smaller quanta (preemptive)
- Processes that use entire quanta are too high priority, bump down to lower priority queue
- Processes that don't use entire quanta are too low priority and moved up to a higher priority queue

Multi Level Feedback

- parameters
- ◊ number of queues
- ◊ the scheduling algorithm for each queue
- ◊ when to upgrade a process
- ◊ when to downgrade a process
- ◊ how to choose the initial queue

- most complex algorithm, is approximated using priorities

Scheduling Algorithms

- FIFO - non preemptive
- SJF - non-preemptive (exponential average)
- SRTF - preemptive
- priority - preemptive/non-preemptive
  ◊ aging
- Round Robin - preemptive (quantum)
- Multiple queues
  ◊ multiple scheduling algorithms
  ◊ mutli-level feedback

Question - In Round Robin scheduling with a quanta of 1/10 second is it possible that more than 10 processes can execute a burst in a given second? Why?

Multiple Processors

- Scheduling is more complex
- ◊ usually a common queue for all processors (load sharing)
- ◊ sometimes hardware limitations (I/O)
- ◊ actual parallel system, have to watch access to kernel data structures such as PCBs and Queues.

- Homogenous/memory sharing processors

- Symmetric / Asymmetric

Real Time Scheduling

- Hard Real Time
  - ◊ guaranteed completion times
  - ◊ resource reservation
  - ◊ dedicated hardware
- Soft Real Time
  - ◊ performance concerns
  - ◊ multimedia
  - ◊ priority scheduling required
  - ◊ low dispatch latency required!!
  - ◊ kernel preemption points
  - ◊ kernel preemptable

Algorithm Evaluation

- Earlier, we talked about criteria

◊ decide on relative importance of each criteria

- CPU utilization – keep CPU as busy as possible
- Throughput – # of jobs done per time unit
- Turnaround Time – Time of submission to Time of Completion
- Waiting Time – amount of time in ready queue
- Response Time – submit time to time of first output request

Algorithm Evaluation

- Deterministic Modeling

◊ take an example representative workload
- a set of cpu burst times, usually more than one burst time for each process

◊ calculate each of the criteria for each of the algorithms (wait time, turn around time, etc.)

◊ in general, makes too many assumptions

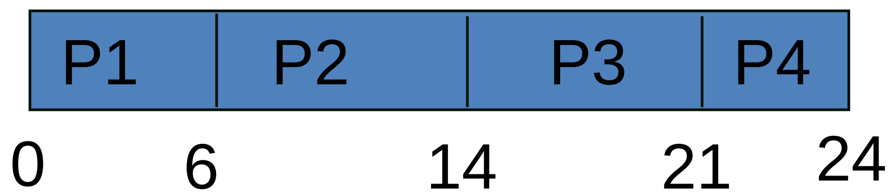
Algorithm Evaluation

- Deterministic Modeling

◊ Gantt charts

P1 - 6ms, P2 - 8ms, P3 - 7 ms, P4 - 3 ms

What is total & average waiting time with FIFO scheduling.



- Waiting time:

  - P1 - 0ms, P2 - 6ms, P3 - 14 ms, P4 - 21 ms = 41ms

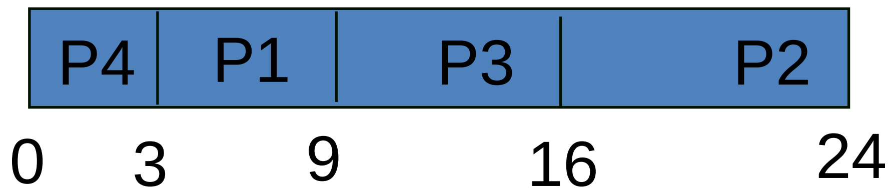
- average = 10.25ms

Algorithm Evaluation

- Same processes

P1 - 6ms, P2 - 8ms, P3 - 7 ms, P4 - 3 ms

What is total & average waiting time with SJF scheduling.



- Waiting time:

  - P1 - 3ms, P2 - 16 ms, P3 - 9 ms, P4 - 0 ms = 28ms

- average = 7ms

Algorithm Evaluation

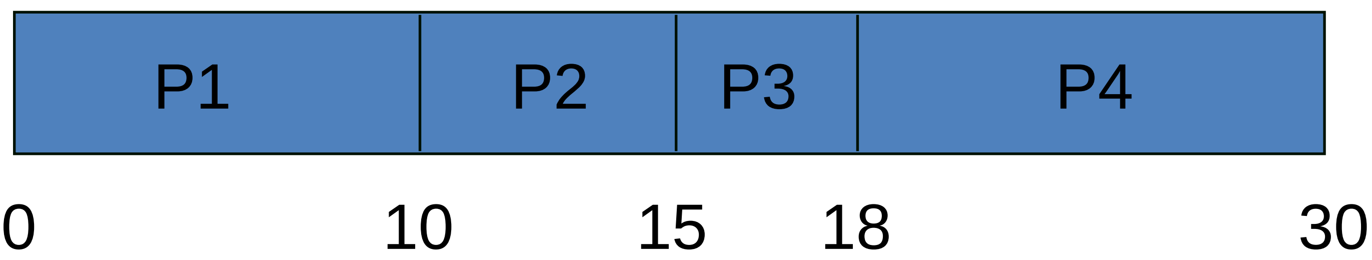
- Simulation
  - ◊ simulate all of the relevant parts of the system
  - ◊ difficult to link various parts of the model
  - ◊ trace tapes (generated from real systems)

- Implementation
  - ◊ try it and find out.
  - ◊ expensive

Scheduling Examples

P1 - 10 ms, P2 - 5 ms, P3 - 3 ms, P4 - 12 ms

FIFO



Wait Times:

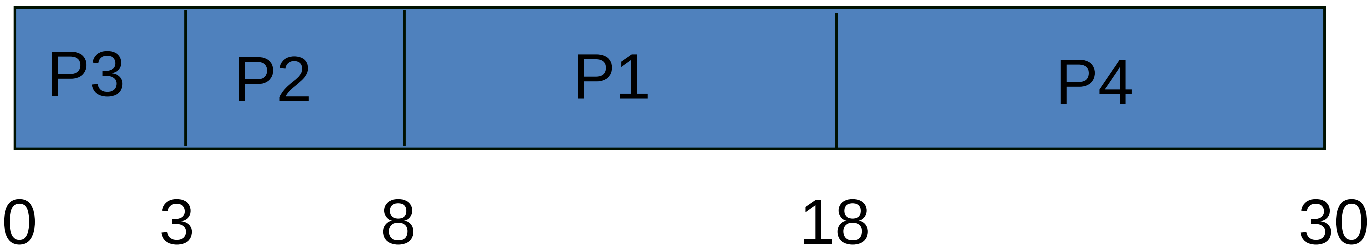
P1: 0 P2: 10 P3: 15 P4: 18

Total: 43 Average: 10.75

Scheduling Examples

P1 - 10 ms, P2 - 5 ms, P3 - 3 ms, P4 - 12 ms

SJF



Wait Times:

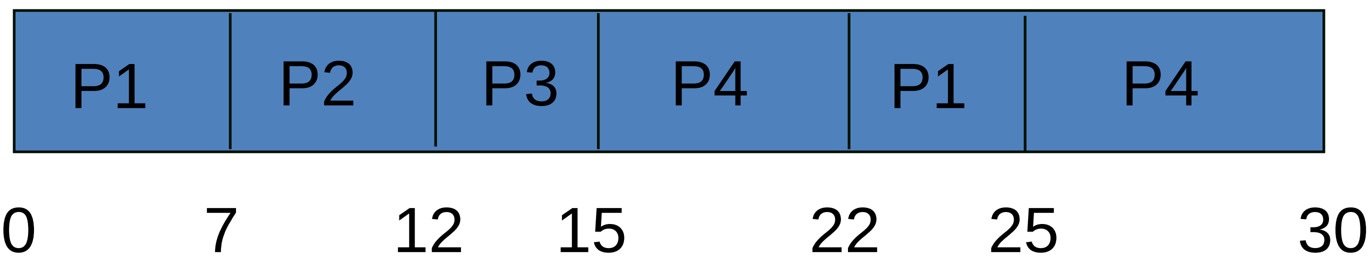
P1: 8 P2: 3 P3: 0 P4: 18

Total: 29 Average: 7.25

Scheduling Examples

P1 - 10 ms, P2 - 5 ms, P3 - 3 ms, P4 - 12 ms

RR, q=7ms, no context overhead



Wait Times:

P1: (22-7) = 15 P2: 7 P3: 12 P4: 15+(25-22)= 18

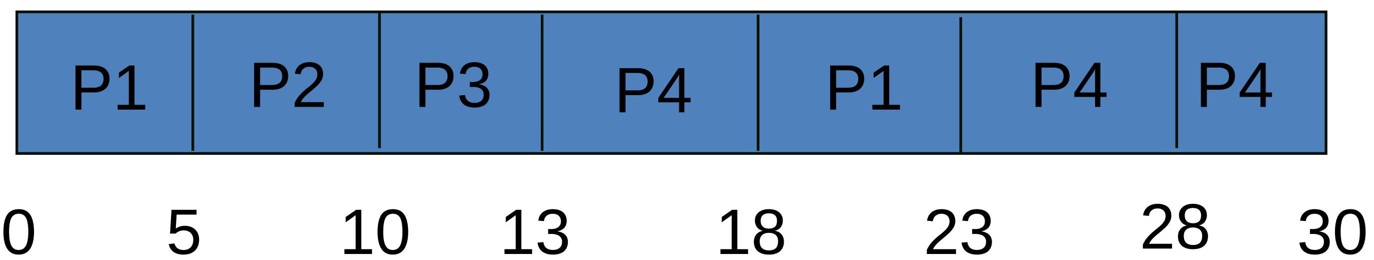
Total: 52 Average: 13

Turnaround for P2: 12 P3: 15

Scheduling Examples

P1 - 10 ms, P2 - 5 ms, P3 - 3 ms, P4 - 12 ms

RR, q=5ms, no context overhead



Wait Times:

P1: 13 P2: 5 P3: 10 P4: 18

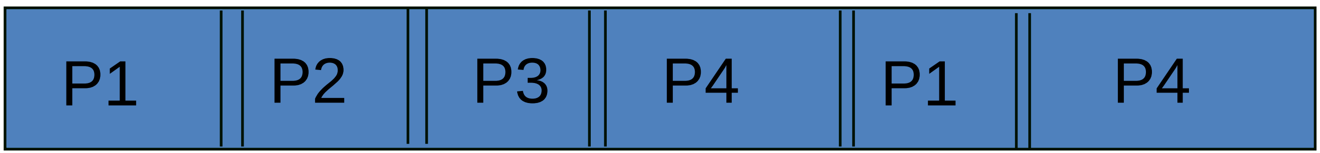
Total: 46 Average: 11.5

Turnaround for P2: 10 P3: 13

Scheduling Examples

P1 - 10 ms, P2 - 5 ms, P3 - 3 ms, P4 - 12 ms

RR, q=7ms, 1 ms context overhead



0 7,8 13,14 17,18 25,26 29,30 35

Wait Times:

P1: (26-7)=19 P2: 8 P3: 14 P4: 18+(30-25)=23

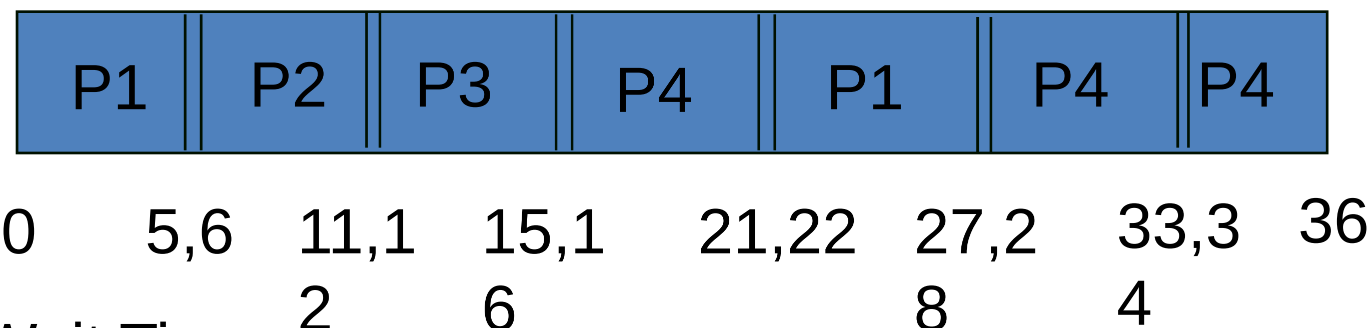
Total: 64 Average: 16

Turnaround for P2: 13 P3: 17

Scheduling Examples

P1 - 10 ms, P2 - 5 ms, P3 - 3 ms, P4 - 12 ms

RR, q=5ms, 1 ms context overhead



Wait Times:

P1: 17 P2: 6 P3: 12 P4: 24

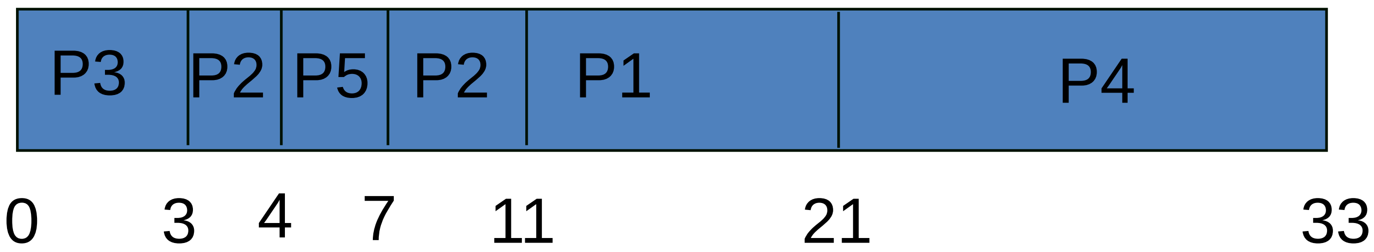
Total: 59 Average: 14.75

Turnaround for P2: 11 P3: 15

Scheduling Examples

P1 - 10 ms, P2 - 5 ms, P3 - 3 ms, P4 - 12 ms

SRTF, interrupt at time 4, P5 - 3 ms



Wait Times:

P1: 11 P2: 6 P3: 0 P4: 21 P5: 0

Total: 38 Average: 7.6