do questions from text

pg 326

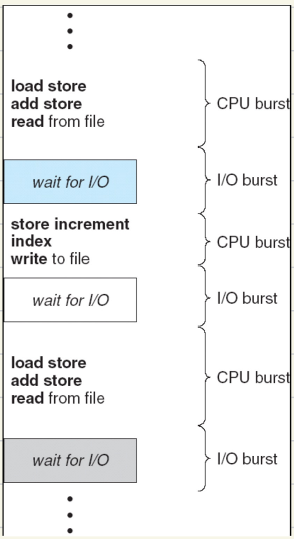
Basic Concepts

- CPU - I/O Bursts : process execution alternates b/w CPU and I/O ops.

CPU burst ⇒ duration of 1 CPU exe. cycle
I/O burst ⇒ duration of 1 I/O operation (wait time)

- CPU scheduling objective : keep the CPU busy as much as poss.

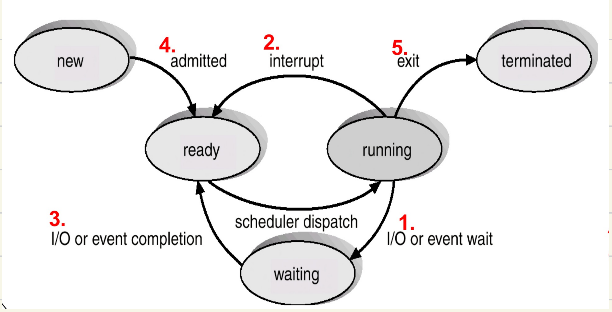
focussing on concepts of short-term (ready queue) scheduler for uni and multiprocessor CPUs



one cycle

CPU Scheduler (Short Term)

- goal : select process (1/t) in the ready queue and allocate it to the CPU (1 process/core)
- the scheduler runs whenever the process changes state



- ready → running is called scheduler dispatch

  always made by the CPU-scheduler

- the CPU scheduler has to execute when a process is in the exit or event / I/O wait, else the CPU will idle

Types of CPU scheduler

1. Nonpreemptive — once a process has been assigned to the CPU, it keeps it until it voluntarily releases the CPU either by
   a) terminating
   b) I/O or event wait

↓

scheduling is nonpreemptive if it happens only for exit or I/O/event wait (and occasionally admission when core is idle)

2. Preemptive : CPU can be taken away from a running process at any time

↓

can be for 1, 5 and 4 but ALSO for 2, 3, 4 or at any other time instant

Scheduling Objectives

SYSTEM-WIDE

1. Max CPU utilization → measured by

$$\frac{\text{total exe time on each core}}{\text{total time} \times \text{no of cores}}$$

2. Max throughput → number of processes completing exe/time
   or
   no. of exit transitions per unit time

1. Min. Turnaround Time : amount of time taken to execute a process

↳ from "admitted" to "exit"

average over all processes is denoted as average turnaround time

(state) (named time)
running (CPU burst)
+ waiting (I/O burst)
+ ready (waiting time)

turnaround time

2. Min waiting time : amount of time a process has been waiting in ready state

NOT THE TIME IN THE WAIT STATE

3. Min response time : time from a request submission (admitted) to response produced (assumed to coincide with start of execution)

PROCESS-SPECIFIC

assume a single CPU burst, single CPU core per process.
focus on average waiting time

1. FCFS
2. SJF
3. Priority Based Scheduling
4. Round Robin
5. Multilevel Queue Scheduling

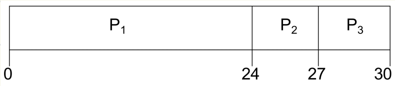
Scheduling Algorithms : Uniprocessor System

1. First Come First Serve Scheduling

- as the name suggests.
- can be implemented using a FIFO queue

eg.

Process	CPU Burst Length
P_1	24
P_2	3
P_3	3



→ all arrive at 0, in the order P1, P2, P3

waiting time : $P_1 \to 0$
$P_2 \to 24$
$P_3 \to 27$

avg $= (0+24+27)/3 = 17$

since the process keeps the CPU until termination / wait
this is non-preemptive.

if they arrive in the order P_2, P_3, P_1 :
waiting time $P1 \to 6$
$P2 \to 0$
$P3 \to 3$

avg $= (6+3+0)/3 = 3$

clearly, average waiting time is affected by arrival order

PROPERTIES

- nonpreemptive : processes have to voluntarily release the CPU once allocated
- convoy effect : short processes suffer long waiting times if they arrive after a long process

2. Shortest Job First (SJF) Scheduling

- prioritizing processes based on their CPU burst lengths
  - shorter burst ⇒ higher priority.
  - handles convoy effect of FCFS

1. Nonpreemptive : once the core is given to a process, it stays until the CPU burst is completed

2. Preemptive / Shortest Remaining time first : if a newly created process has CPU burst length less than the remaining CPU burst of a currently running process, then preemption will occur and it will get switched out.

SJF / SRTF is optimal for optimizing avg waiting time

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

non.prem.

P1	P3	P2	P4

0 7 8 12 18

P2 (6) P3 (3) P4 (7)

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

pre-emp.

P1	P2	P3	P2	P4	P1

0 2 4 5 7 11 16

5 left (11-2=9) 2 left (5-4=1) P4 4 left (7-5=2)

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

3. Priority-based Scheduling

- a priority number (interger) is associated with each process
  - ↳ smallest number ⇒ greatest priority
- FCFS & SJF are forms of this
  - FCFS ↓ arrival time
  - SJF ↓ CPU burst length
- Problem of starvation : lower priority may never execute
  - ↓ solution : Aging
    - ↓ as time progresses, system gradually increases priority

4. Round Robin (RR) Scheduling

- a "fair scheduling algorithm"
- all processes get the CPU for q time units; pre-empted (timer interrupt) after and placed in the ready queue
- q = quantum time (normally 10-100 milliseconds)
- for n processes in the ready queue ⇒ waiting time is not more than $(n-1)q$
- large q (greater than CPU burst) → becomes FCFS
- small q (many context switches, high overhead)
- higher avg waiting/turnaround time than SJF, but better response time.
- no well def. relation b/w avg turnaround time & quantum size although ofc: avg turnaround time is fixed once quantum size is greater than the max CPU burst length

5. Multilevel Queue Scheduling

- diff processes have different requirements and hence need different scheduling
  ↓

  so the ready queue is partitioned into separate queues and each queue has its own scheduling algorithms.

- foreground processes : keystrokes, I/O interaction need to be interactive → fast response time so RR is pref.

- background processes : batch processing, virus scanning need not be interactive → FCFS.

- each queue has its own scheduling and the queues themselves need to be scheduled to.

fixed priority	time slice based
- serve all foreground queues first, then after serving all processes, consider the background queue - starvation for lower priority queue	- each queue gets a fixed time quantum on the CPU (80ms fore, 20ms back.)

1. Partitioned Scheduling

2. Global Scheduling

Scheduling Algorithms : Multiprocessor System

ASSUME multi-core CPU, each process can execute on only one CPU-core at any time instant.

1. Partitioned scheduling (Asymmetric Multiprocessing AMP)

- Processes are partitioned apriori (based on theoretical deduction) among the CPU cores (at process creation time)

→ each process is mapped to one core
→ separate uniprocessor CPU scheduling for each core

- several queues, specific to cores.

advantages

- core-specific scheduling strategy is feasible (FCFS, STF, RR, multilevel)
- easy and simple extension of uniprocessor CPU scheduling to multiprocessor CPU scheduling

disadvantages

- mapping is tough, the load must ideally be balanced else cores could be idle while others are overloaded
- kind of like knapsack problem
- heuristic such as best-fit can be used to assign the process to the CPU

2. Global Scheduling (Symmetric Multiprocessing)

- one or more ready queues for the whole system (no more matching)
- when any core becomes idle, CPU scheduler runs and allocates the next process to execute
- process selection based on strategy applied globally or symmetrically across all cores (FCFS, SJF, RR)
- the same process could execute on different cores at different time
- pre-empted.

disadvantages

- implementation complexity is high compared to partitioned scheduling
  - ↳ the cores need to be synced, scheduler must use 1 clock.
  - ↳ selection of next process may involve going through several processes in the ready queue
- switching cores means loss of access to prev cores data cache, so data needs to migrate which is time consuming
- overhead is high