Kamesh Vedula 68149804

CS 143A: Principles of Operating Systems

Homework #2

Due Date: Fri, 4 May 2018, 11:55 PM

Please submit a *PDF file* containing all answers to EEE+ Canvas. Total Marks=100

Question 1: CPU Scheduling [5, 20, 20]

a. [5 pts]) For each of the following scheduling algorithms, state whether it could result in starvation and (briefly) explain your answer:

1. First-Come First-Served (FCFS)
2. Shortest Job First (SJF) (Non-preemptive) if short jobs are continuosly added.
3. Shortest Remaining Time First (SRTF) (Preemptive) if short jobs are continuosly added.
4. Round Robin (RR)
5. Priority if priority jobs are continuosly added. won't let low priority jobs execute.

b. [20 pts]) Consider the set of process:

Process ID	Arrival Time	Burst Time	Priority
P1	0	20	4
P2	9	40	2
P3	15	10	3
P4	25	30	1

~~20~~ ~~10~~ 0

~~40~~ ~~30~~ ~~20~~ ~~10~~ 0 ← work for RR

~~10~~ 0

~~30~~ ~~20~~ ~~10~~ 0

PQ

~~P1~~
~~P2~~
P1
~~P3~~
~~P2~~
~~P4~~
~~P2~~
~~P4~~
~~P2~~
P4

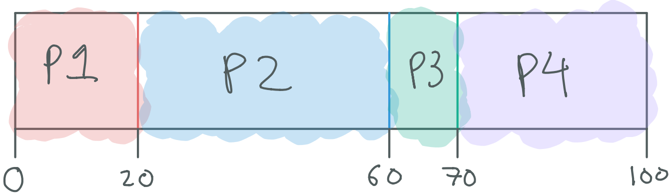
Draw the GANTT chart for the following scheduling algorithms. Break ties according to process ID e.g. P8 takes precedence over P9 if they're equivalent according to the algorithm.

1. First-Come First-Served (FCFS)
2. Shortest Job First (SJF) (Non-preemptive)
3. Round Robin (RR) (Time Quantum = 10)
4. Priority (preemptive)

P1	P2	P1	P3	P2	P4	P2	P4	P2	P4

0 10 20 30 40 50 60 70 80 90 100

FCFS



Avg. Waiting Time $= (0+11+45+45)/4 = 25.25$

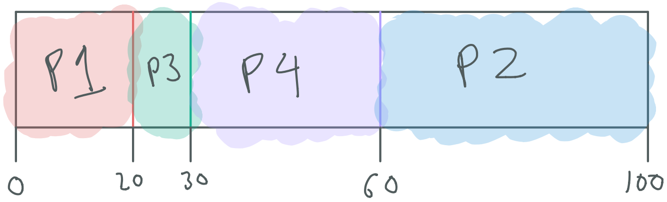
P1 : $0-0=0$ P3 : $60-15=45$

P2 : $20-9=11$ P4 : $70-25=45$

Avg. TAT

$\left(\begin{array}{l} 0+20 \\ +11+40 \\ +45+10 \\ +45+30 \end{array}\right) \div 4 = 50.25$

SJF



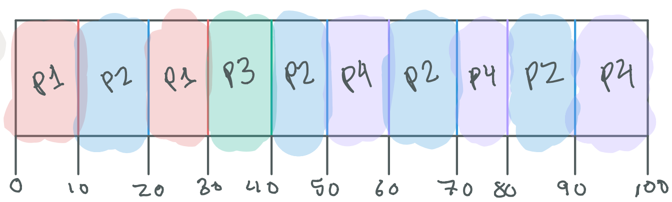
Avg. Waiting Time $= (0+5+5+51)/4 = 15.25$

P1 : $0-0=0$ P3 : $20-15=5$

P2 : $60-9=51$ P4 : $30-25=5$

$\left(\begin{array}{l} 0+20 \\ +51+40 \\ +5+10 \\ +5+30 \end{array}\right) \div 4 = 40.25$

RR



Avg. Waiting Time $= (10+50+30+70)/4 = 40$

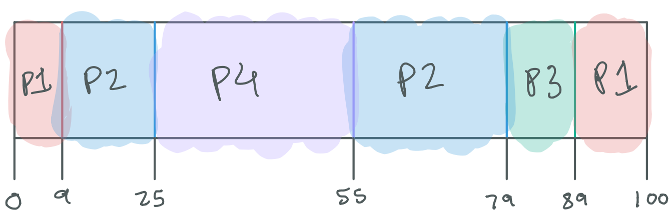
P1 : $20-10=10$ P3: 30

P2 : $10+(40-20)+(60-50)+(80-70)$ $= 10+20+10+10$ $= 50$

P4 : $50+(70-60)+(90-80)$ $= 50+10+10$ $= 70$

$\left(\begin{array}{l} 10+20 \\ +50+40 \\ +30+10 \\ +70+30 \end{array}\right) \div 4 = 65$

P



Avg. Waiting Time $= 80+39+79+25/4 = 55.75$

P1 : $9+(89-9)=80$ P3 : 79

P2 : $9+(55-25)$ $=39$ P4 : 25

$\left(\begin{array}{l} 100+ \\ 79-9+ \\ 89-15+ \\ 55-25 \end{array}\right) \div 4 = 68.5$

c. [20 pts]) Complete the following table according to your Gantt charts above. Note that you will receive *partial credit* if you made a mistake with the Gantt charts but calculate the following metrics correctly.

Scheduling Algorithm	Average waiting time (ms)	Average turnaround time (ms)
First Come First Served (FCFS)	25.25	50.25
Shortest Job First (SJF) (Non-preemptive)	15.25	40.25
Round Robin (RR) (Time Quantum = 10)	40	65
Priority (preemptive)	55.75	68.5

Question 2: Critical Sections [30]

Consider the following algorithm that provides a 2-process solution to the critical section problem:

flag[0] = false; flag[1] = false;	

P0:
while (true) {
flag[0] = true;
while (flag[1]) **{**
flag[0] = false;
while (flag[1]) **{**
no-op;
}
flag[0] = true;
}
critical section
flag[0] = false;
remainder section
}
	P1:
	while (true) {
	flag[1] = true;
	while (flag[0]) **{**
	flag[1] = false;
	while (flag[0]) **{**
	no-op;
	}
	flag[1] = true;
	}
	critical section
	flag[1] = false;
	remainder section
	}

a. [30 pts]) Specify which of the following requirements are satisfied or not by this algorithm. Explain why or why not.

1. Mutual Exclusion
2. Progress
3. Bounded Waiting

Mutual Exclusion: Satisfied Both check each others flags and flips their flag if the other is in progress, thereby keeping only one process in the critical section at a time.

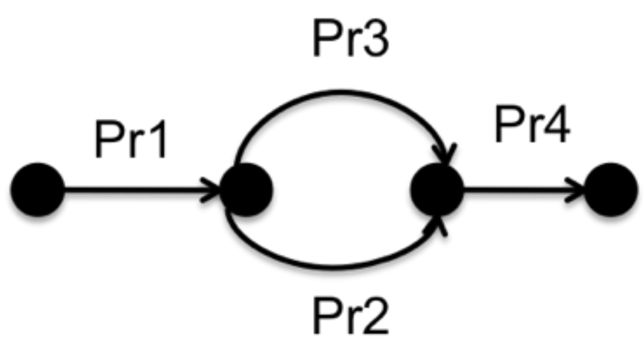
Progress: Not Satisfied Algorithm can't progress if both flags are set to true at the same time.

Bounded Wait: Not Satisfied Process (P0) can loop forever.

Question 3: Semaphores [5, 20]

In an operating system processes can run concurrently. Sometimes we need to impose a specific order in execution of a set of processes. We represent the execution order for a set of processes using a process execution diagram.

Consider the following process execution diagram. The diagram indicates that **Pr1** must terminate before **Pr2**, **Pr3** and **Pr4** start execution. It also indicates that **Pr4** should start after **Pr2** and **Pr3** terminate and **Pr2** and **Pr3** can run concurrently.



We can use semaphores in order to enforce the execution order. Semaphores have two operations:

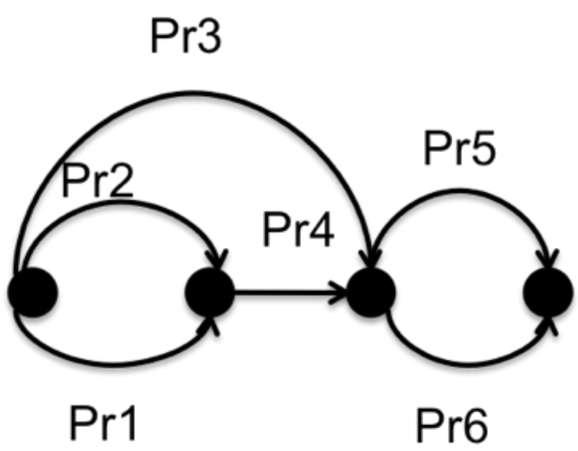
- **P** (or wait) is used to acquire a resource. It waits for semaphore to become positive, then decrements it by 1.
- **V** (or signal) is used to release a resource It increments the semaphore by 1, waking up the blocked processes, if any.

The above process execution diagram can be represented using **Serial** and **Parallel** notation.
The above execution diagram is represented as **Serial(P1, Parallel(P2, P3) , P4)**.
We can use the following semaphores to enforce the execution order above:

s1=0; s2=0; s3=0;
Pr1: body; V(s1); V(s1);
Pr2: P(s1); body; V(s2);
Pr3: P(s1); body; V(s3);
Pr4: P(s2); P(s3); body;

Assume that the semaphores **s1**, **s2**, and **s3** are created with an initial value of **0** before processes **Pr1**, **Pr2**, **Pr3**, and **Pr4** execute.

Based on this explanation, consider the following process execution graph and answer the questions below:



a. [5 pts]) Write the execution process of the processes using **Serial** and **Parallel** notation.

Serial (Parallel (P3, Serial (Parallel (P2, P1), P4)), Parallel (P5, P6))

b. [20 pts]) Using semaphores enforce execution order according to the process execution diagram.

s1 = 0; s2 = 0; s3 = 0; s4 = 0;

Pr 1: body ; V(s1);

Pr 2: body; V(s2);

Pr 3: body; V(s3);

Pr 4: P(s1); P(s2); body; V(s4);

Pr 5: P(s4); P(s3); body;

Pr 6: P(s4); P(s3); body;