CS205-Operating Systems

Saturday Apr 07, 2018

Course Instructor

Rizwan Ul Haq, Dr. Rana Asif Rehman, Muhammad Tayyab

Serial No:

2nd Midterm Exam

Total Time: 1 Hour

Total Marks: 55

____
Signature of Invigilator

Roll No ______________ Section ______________ Signature ______________

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Verify at the start of the exam that you have a total of Four (4) questions printed on Seven (7) pages including this title page.
2. Attempt all questions on the question-book.
3. The exam is closed books, closed notes. Please see that the area in your threshold is free of any material classified as 'useful in the paper' or else there may a charge of cheating.
4. Read the questions carefully for clarity of context and understanding of meaning and make assumptions wherever required, for neither the invigilator will address your queries, nor the teacher/examiner will come to the examination hall for any assistance.
5. Fit in all your answers in the provided space. You may use extra space on the last page if required. If you do so, clearly mark question/part number on that page to avoid confusion.
6. Use only your own stationery and calculator. If you do not have your own calculator, use manual calculations.
7. Use only permanent ink-pens. Only the questions attempted with permanent ink-pens will be considered. Any part of paper done in lead pencil cannot be claimed for checking/rechecking.

	Q-1	Q-2	Q-3	Q-4	Total
Total Marks	18	10	7	20	**55**
Marks Obtained					

Vetted By: ______________________________ **Vetter Signature:** ______________________

Q1.	4 + 4 + 4 + 4 + 2 = 18

Give precise answers in the space provided.

a. Briefly explain the difference between shared memory and message passing? In which circumstances shared memory is given preference over message passing? 2 + 2

In the shared-memory model, a region of memory that is shared by cooperating processes is established. Processes can then exchange information by reading and writing data to the shared region. In the message-passing model, communication takes place by means of messages exchanged between the cooperating processes.
In shared memory access to shared memory segment is no more monitored by kernel and the sharing processes mange it themselves. On the other hand in message passing all the communication go through the kernel and is handled by the kernel itself.
Shared memory is much faster than message passing and if large amount of data needs to be shared then shared memory is preferred over message passing.

b. Which of the following components of program state are shared across threads in a multithreaded process? 1*4

Component	Answer (Yes/No)
Register Values	No
Heap Memory	Yes
Global Variables	Yes
Stack Memory	No

c. Provide two programming examples in which multithreading does not provide better performance than a single-threaded solution? 2 + 2

(1) Any kind of sequential program is not a good candidate to be threaded. An example of this is a program that first downloads a file and then print it after complete download. Making two thread to download and print will not be of any use.

(2) Another example is a "shell" program such as the C-shell or Korn shell. Such a program must closely monitor its own working space such as open files, environment variables, and current working directory

d. How named pipes are different from ordinary pipes? Explain each with the help of example. 2 + 2

Ordinary pipes allow two processes to communicate in standard producer-consumer fashion. Ordinary pipes are unidirectional, allowing only one-way communication. Example for ordinary pipe is a file descriptor with two ends in a child and parent process, "dir | more" and "ls | more" commands.
Named pipes provide a much more powerful communication tool. Communication can be bidirectional, and no parent–child relationship is required. Once a named pipe is established, several processes can use it for communication. Named pipes continue to exist after communicating processes have finished. SQL server is an example of named pipe.

e. When using remote procedure call an issue may occur that the data representation on the client and server machines are different. How is this problem resolved in Remote Procedure call? 2

External data format XDR is used to send data.

Q2.	**3 + 3 + 4 = 10**

What are three models of mapping user level threads to kernel level threads? List and draw diagram. Also discuss one pros and cons of these three.

1. **Many-to-One:** Thread management is done by the thread library in user space, so it is efficient. However, the entire process will block if a thread makes a blocking system call

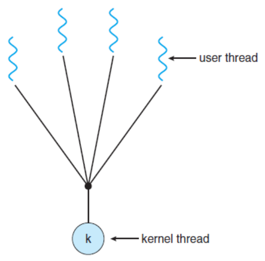


Figure 4.5 Many-to-one model.

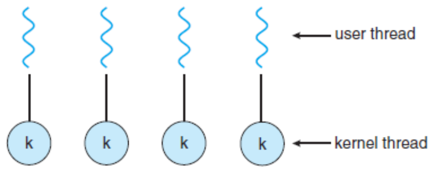


Figure 4.6 One-to-one model.

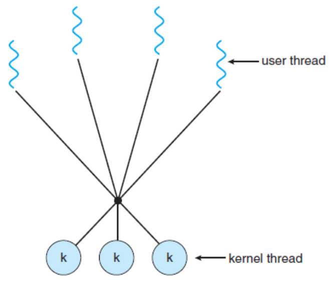


Figure 4.7 Many-to-many model.

2. **One-to-One:** It provides more concurrency than the many-to-one model by allowing another thread to run when a thread makes a blocking system call. It also allows multiple threads to run in parallel on multiprocessors. The only drawback to this model is that creating a user thread requires creating the corresponding kernel thread.

3. Many-to-Many: This model is free of the problems of many-to-one and one-to-one model.

Q3.	**3 + 4 = 7**

Briefly discuss the following issues of multithreading?

a. The fork() and exec() System Calls mechanism

If one thread in a program calls fork(), does the new process duplicate all threads, or is the new process single-threaded? Some UNIX systems have chosen to have two versions of fork(), one that duplicates all threads and another that duplicates only the thread that invoked the fork() system call.
The exec() system call typically works in the same way as in single threaded environment. That is, if a thread invokes the exec() system call, the program specified in the parameter to exec() will replace the entire process—including all threads.

b. Signal Handling:

Delivering signals is more complicated in multithreaded programs, where a process may have several threads. Where, then, should a signal be delivered? In general, the following options exist:

1. Deliver the signal to the thread to which the signal applies.
2. Deliver the signal to every thread in the process.
3. Deliver the signal to certain threads in the process.
4. Assign a specific thread to receive all signals for the process

Synchronous signals need to be delivered to the thread causing the signal and not to other threads in the process. However, the situation with asynchronous signals is not as clear. Some asynchronous signals—such as a signal that terminates a process (<control><C>, for example)—should be sent to all threads.

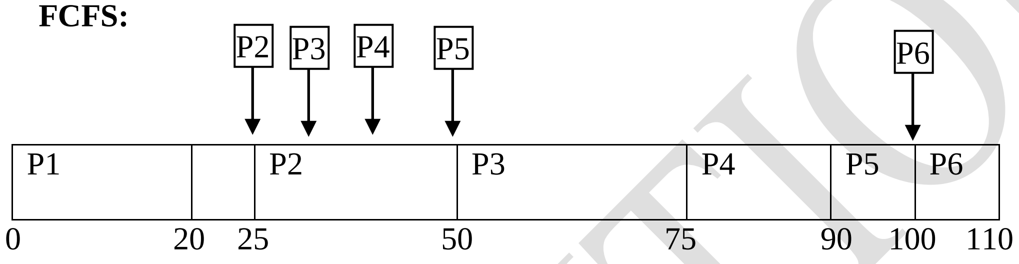
Most multithreaded versions of UNIX allow a thread to specify which signals it will accept and which it will block. Therefore, in some cases, an asynchronous signal may be delivered only to those threads that are not blocking it.

Q4.	**5 + 5 + 5 + 5 = 20**

Calculate the **average waiting time** and **average turnaround time** of the processes using FCFS, SJF and SRTF and RR for the following set of processes. Draw Gantt chart clearly. For RR use Time Quantum = 10

Process	Burst	Arrival Time
P1	20	0
P2	25	25
P3	25	30
P4	15	40
P5	10	50
P6	10	100

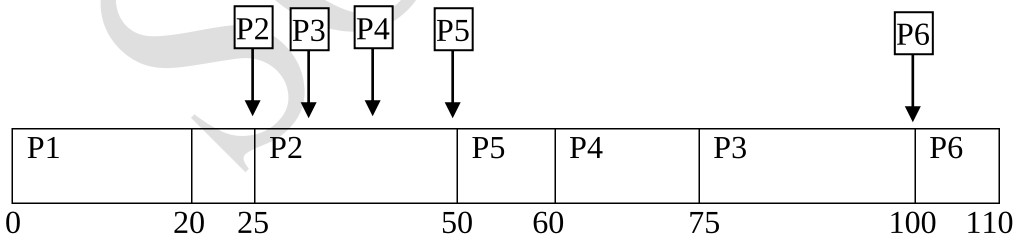
FCFS:



Waiting Time			Turn Around Time	
P1	0		P1	20-0 = 20
P2	0		P2	50-25 = 25
P3	75-30-25 = 20		P3	75-30 = 45
P4	90-40-15 = 35		P4	90-40 = 50
P5	100-50-10 = 40		P5	100-50 = 50
P6	110-100-10 = 0		P6	110-100 = 10

Average Waiting Time = (0 + 0 + 20 + 35 + 40 + 0) / 6 = 15.83
Average Turnaround Time = (20 + 25 + 45 + 50 + 50 + 10) / 6 = 33.33

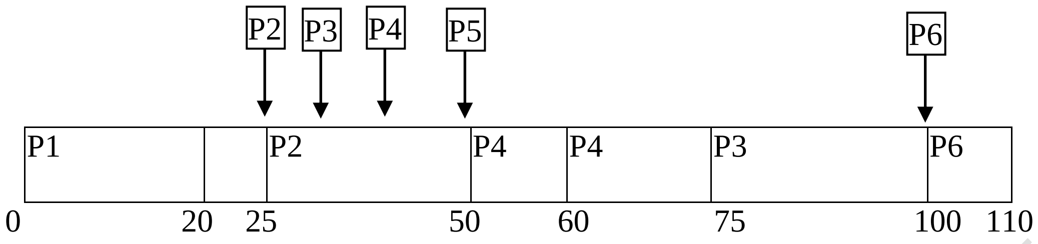
Shortest Job First:



Waiting Time			Turn Around Time	
P1	20-0-20 = 0		P1	20-0 = 20
P2	50-25-25 = 0		P2	50-25 = 25
P3	100-30-25 = 45		P3	100-30 = 70
P4	75-40-15 = 20		P4	75-40 = 35
P5	60-50-10 = 0		P5	60-50 = 10
P6	110-100-10 = 0		P6	110-100 = 10

Average Waiting Time = (0 + 0 + 45 + 20 + 0 + 0) / 6 = 10.83
Average Turnaround Time = (20 + 25 + 75 + 35 + 10 + 10) / 6 = 28.33

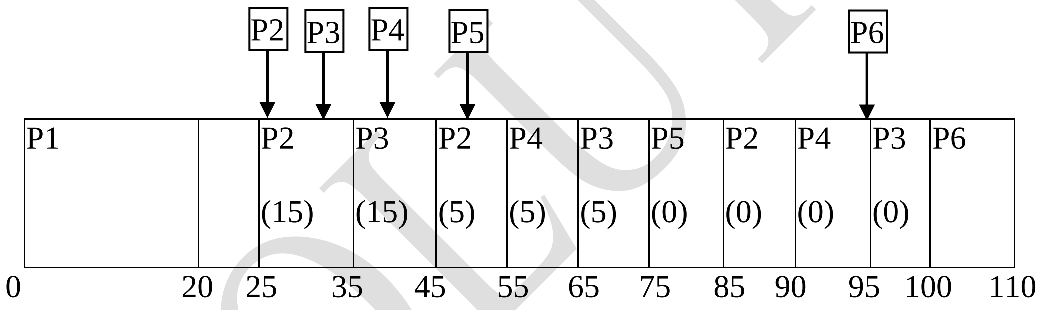
Shortest Remaining Time First:



Waiting Time			Turn Around Time	
P1	20-0-20 = 0		P1	20-0 = 20
P2	50-25-25 = 0		P2	50-25 = 25
P3	100-30-25 = 45		P3	100-30 = 70
P4	75-40-15 = 20		P4	75-40 = 35
P5	60-50-10 = 0		P5	60-50 = 10
P6	110-100-10 = 0		P6	110-100 = 10

Average Waiting Time = (0 + 0 + 45 + 20 + 0 + 0) / 6 = 10.83
Average Turnaround Time = (20 + 25 + 70 + 35 + 10 + 10) / 6 = 28.33

Round Robin:



Waiting Time			Turn Around Time	
P1	20-0-20 = 0		P1	20-0 = 20
P2	90-25-25 = 40		P2	90-25 = 65
P3	100-30-25 = 45		P3	75-30 = 45
P4	95-40-15 = 40		P4	95-40 = 55
P5	85-50-10 = 25		P5	85-50 = 35
P6	110-100-10 = 0		P6	110-100 = 10

Average Waiting Time = (0 + 40 + 45 + 40 + 25 + 0) / 6 = 25.00
Average Turnaround Time = (20 + 65 + 45 + 55 + 40 + 10) / 6 = 38.33