OSN ‘24 Quiz 1 Key

Q1

- Mentioning states of process (without reason): 2 marks
  - -0.5 for every wrong entry in the above table (note that the answer is not required to have a table, it could be in the form of a paragraph as written above).
- Mentioning reasons for all states: 2 marks
  - -0.5 for every wrong/missing reason for any timestep

Set A

Process 1 arrives at t = 0 and starts running. Process 2 arrives at time t = 10 sec. At time t=30, process 1 makes a system call. At t=40, process 1 has completed the system call. At t=50, process 2 completes execution and Process 3 arrives at t=55. At t=60 process 1 completes execution and Process 3 completes execution at t=65.

Time	Process 1	Process 2	Process 3
0	Running	Yet to arrive	Yet to arrive
10	Running	Ready	Yet to arrive
30	Blocked	Running	Yet to arrive
40	Ready	Running	Yet to arrive
50	Running	Completed	Yet to arrive
55	Running	Completed	Ready
60	Completed	Completed	Running
65	Completed	Completed	Completed

Process 1 starts running at time t=0. When process 2 arrives at t=10, process 1 continues running as we see it making a system call at time t=30 (which can only be done if it’s running). Process 1 goes to blocked state while the only other ready process (Process 2) starts running. At time t=40, process 1 completes it’s I/O and goes back to ready state. However, we can deduce that Process 2 keeps running instead of Process 1 here as it completes before

Process 1 at t=50. Process 1 also gets the CPU back and starts running at t=50. Process 3 arrives at t=55 but waits for the CPU in ready state as we have deduced that the system is non-preemptive previously. After process 1 completes at t=60, process 3 starts running and finally completes at t=65.

Set B

> Process x arrives at t = 0 and starts running. Process y arrives at time t = 10. At time t=5, process x makes a system call. At t=15, process x has completed the system call. At t=30, process y completes execution. Process z arrives at t=35 and at t=40 process x completes execution. Process z completes execution at t=45.

Time	Process x	Process y	Process z
0	Running	Yet to arrive	Yet to arrive
5	Blocked	Yet to arrive	Yet to arrive
10	Blocked	Running	Yet to arrive
15	Ready	Running	Yet to arrive
30	Running	Completed	Yet to arrive
35	Running	Completed	Ready
40	Completed	Completed	Running
45	Completed	Completed	Completed

Process x arrives at t=0 and starts running as CPU is free. It makes an I/O call and goes into blocked state at t=5. Process y arrives at t=10 and starts running as the CPU is free at the time. After process x finishes I/O at t=15, it goes into Ready state waiting for process y to finish executing. We deduce this by observing that process y finishes before process x at time t=30, hence the algorithm used for scheduling may be non-preemptive. Process x starts running after CPU is done with process y at t=30 too. At t=35, process Z comes into the queue and is ready to run but doesn't until t=40 due to non-preemptive nature. At t=45, all processes have completed their execution.

Set C

Process p1 arrives at t = 0 and starts executing. Process p2 arrives at time t = 15 sec. At time t=25, process p1 makes a system call. At t=35, process p1 has completed the system call. At t=45, p2 completes execution and process, p3 arrives at t=50 and completes execution at t=60. Process p1 completes execution at t=55.

Time	Process p1	Process p2	Process p3
0	Running	Yet to arrive	Yet to arrive
15	Running	Ready	Yet to arrive
25	Blocked	Running	Yet to arrive
35	Ready	Running	Yet to arrive
45	Running	Completed	Yet to arrive
50	Running	Completed	Ready
55	Completed	Completed	Running
60	Completed	Completed	Completed

Process p1 arrives at t=0 and starts executing as the CPU is free. Process 2 arrives at t=15 and stays in the Ready state as the scheduling algorithm seems to be non-preemptive. We deduce that it's non-preemptive because p1 makes an I/O call at t=25. Even after p1 completes the system call at t=35, p2 continues executing due to the non pre-emptive scheduling. At t=45, process p2 finishes executing and p1 gets the CPU. At t=50, p3 arrives but waits in the ready state as p1 is already running. At t=55, p1 finishes execution at p3 starts executing. At t=60, p3 finishes execution.

Q2

- Marking correct option: 1.5 marks
- Giving correct reasoning for first part of the question (value of childId/value of PID): 0.5 marks
- Giving correct reasoning for second part of the question (execution order): 1 mark

Set A

What will be value of childId when fork() system call is made and who will get executed first (The parent or the child)? Explain the reason **(3 points)**

A. Child process will get chilId 0, parent will get childId equal to PID of child and parent executes first before child
B. Child process will get childId 0, parent will get childId equal to PID of child and child executes before the parent
C. Child process will get chilldId 0, parent will get childId equal to PID of child and the execution order is not deterministic
D. Child process will get childId equal to PID of the child, parent will get childId value 0 and the parent process completes execution before the child
E. Child process will get childId equal to PID of the child, parent will get childId value 0 and the execution order is non-deterministic

Ans. (C)

Explanation: When forking, the child process always receives a return value of 0 (stored in childId here) and the parent gets the actual pid of the child created by the fork. Further, since there are no mechanisms like wait to ensure which process runs first, the order is non-deterministic. This is so because the child and parent process are two different processes and may be scheduled at any time depending on the scheduler.

Set B

A. The PID of the child will be same as that of the parent and the parent executes first before child
B. The PID of the child will be same as that of the parent and the child executes first before parent
C. Parent process PID will be set to the PID of child, child will get random PID and the execution order is non deterministic
D. Child process will get PID of the parent process and the execution order is non deterministic
E. Child process will be assigned a PID different from the parent process and the execution order is non deterministic

Ans. (E)

Explanation: Fork spawns a new process when creating a child process which will thus, have a different PID from the parent process as PID has to be unique.

Further, since there is no command like wait() used to explicitly define the order of execution, it's non-deterministic here. This is because both the child and parent process are 2 separate processes in the system and can be scheduled at any time depending on the scheduler.

Set C

What will be value of childId when fork() system call is made and what will be the order of execution? Explain the reason **(3 points)**

A. In child process, childId will be equal to PID of parent, parent will get childId 0 and the parent process completes execution before the child
B. Child process will get childId value 0, parent process will get childId equal to the PID of child and child process executes before the parent.
C. For child process, childId will be equal to PID of parent, parent will get childId 0 and the child process completes execution before the parent
D. Child process will get a childId value of 0, parent will get childId equal to PID of child process and the execution order is not deterministic
E. In Child process, childId value will be equal to PID of the child, parent will get childId value as 0 and the execution order is non-deterministic

Ans. (B)

Explanation: When forking, the child process always receives a return value of 0 (stored in childId here) and the parent gets the actual pid of the child created by the fork. However, since wait is explicitly called in the parent process, it guarantees that the parent process will not execute before the child process is done executing. Hence, child process executes before the parent process here.

Q3

Set A

3. The team has put up a scenario where there are many processes that will be running in the system and they want your support on designing a policy that allows the OS to switch between multiple processes. The runtime of the process is not known apriori and the processes can arrive at different time instances. Moreover, all the major process are going to be interactive in nature and hence the total completion time is not significant as long as every process gets a chance to execute. They would like you to suggest a scheduling policy for the given scenario. Explain with an example how such policy can be effective **(4 points)**

- Mentioned Round Robin Policy as correct with the explanation given as all processes get a chance to execute fairly. Total completion time is not significant and we care only about the execution of the process which is done fairly in Round Robin. (2M)
- Relevant example given - (2M)
- If MLFQ mentioned, and priority boosting written give 1M for explanation and 1M for example. Without priority boosting 0.5 Marks for explanation and 0.5 for example since all process may not get a chance to execute. (It says major processes are interactive and not ALL).
- Rest all no marks since FCFS isn't fair for processes coming later, STCF and SJF require completion time which isn't known here.

Set B

The team has put up a scenario where there are many processes that will be running in the system and they want your support on designing a policy that allows the OS to switch between multiple processes. The runtime of the process is not known apriori and the processes can arrive at different time instances. Moreover, some process are going to be interactive in nature and some will be CPU bound hence the total completion time also needs to be considered and every process needs to be given a chance to execute. They would like you to suggest a scheduling policy for the given scenario. Explain with an example how such policy can be effective **(4 points)**

- Mentioned MLFQ since it prioritizes both turnaround time and response time. We consider both of them here since there are both CPU bound and interactive processes (2M for explanation and 2M for relevant example). Priority boosting ensures that no process starves. 0.5 marks deducted from explanation if priority boosting not mentioned.
- No marks for STCF or SJF, runtime not known apriori.
- The reason why FCFS isn't the answer is because the processes don't all arrive at the same time, and the reason why RR isn't the answer is that though response time is good (with a good time slice), the turnaround time tends to be bad (there's a trade-off between the two that's controlled by the time slice itself)

Set C

The team has put up a scenario where there are many processes that will be running in the system and they want your support in designing a policy that allows the OS to switch between multiple processes. The key assumption is that the runtime of the process is known apriori and the processes can arrive at different time instances. Most of the processes are CPU intensive and hence the total completion time is more significant as opposed to the response time and every process needs to get a chance to execute. They would like you to suggest a scheduling policy for the given scenario. Explain with an example how such a policy can be effective **(4 points)**

- Mentioned STCF since it aims to minimise completion time along with managing CPU bound processes (2M for explanation and 2M for relevant example).
- No marks for MLFQ. Here, completion time is the major priority.
- SJF with reason 1 Marks and 1 Marks since it isn't preemptive. Rest examples no marks (reasons in Set B and Set A).

Q4

4. The team has implemented a low level mechanism that follows the traditional UNIX style by which OS can switch between multiple processes. However the team feels that the mechanism does introduce an overhead. In your opinion what is this overhead about. Explain the reason **(3 points)**
    A. It consumes additional CPU time
    B. Requires additional hardware
    C. It stops all the processes
    D. It significantly increases the usage of memory
    E. All of the above

- Marked answer is A and reason is correct 3 Marks.(1+2)
- Marked answer is D and reason is correct 1 mark.(0.5+0.5)
- Marked answers are A and D and reason is correct 2 Marks.(1+1)
- Else 0

Reason for Option A :- Some overhead time occurs whenever OS is switching between multiple processes. This overhead time occurs during context switch. Saving values from executing registers to Kernel stack and restoring values for next process. This take additional CPU time.

Reason for Option B :- Overhead is not introduced because of the additional hardware.

Reason for Option C :- Processes are not stopped. Instead they are in ready state. These are 2 different things.

Reason for Option D :- Values are being saved from registers onto kernel stack. Also restoring values for next process takes some memory. This causes some memory overhead. However, since significant increase in usage does not happen this option is partially correct.

Q5

Set A

the following scenario: A process is executing in a machine A with IP address 192.156.13.21. This process is trying to communicate to another process running in a machine B with IP address 192.156.13.26. The data contains confidential information. What component should be used for transportation and what should the addressing that the OS in machine B should use to ensure that the data is delivered to the correct process. Explain the reason **(3 points)**

A. Switch and port number
B. Switch and mac address
C. Switch, router and port number
D. Switch, router and mac address
E. Hub and port number

Notice that the two machines are on the same network (and subnet, but you are not expected to know that). This is evident from the fact that the first 3 octets are the same for both addresses. As discussed in class, intra-network communication is handled by **switches**. Routers are only needed for messages going outside the network or subnet.

Machine B would use port numbers to determine which process is supposed to receive the data because port numbers act as identifiers for specific applications or services running on a device, allowing data to be directed to the correct service.

Correct answer: A

- 1 mark for marking correct answer
- 2 marks for giving correct reasoning for the correct answer.

Set B

Now that the team has tested you in basic OS concepts on virtualization, the team wants to check your basic networking understanding. To this end, the team presents you with the following scenario: A process is executing in a machine A with IP address 192.156.13.21. This process is trying to communicate to another process running in a machine B with IP address 172.180.85.16. The data contains confidential information. What component should be used for transportation and what should the addressing that the OS in machine B should use to ensure that the data is delivered to the correct process. Explain the reason **(3 points)**

A. Switch, router and port number
B. Hub and port number
C. Switch and port number
D. Switch, router and mac address
E. Switch and mac address

The IPs have no common octets → they are on different networks. As soon as you need to communicate across networks, you need to involve *routers*. Sending a message from A to its gateway (router) and from B's gateway to B requires the use of *switches*.

Machine B would use port numbers to determine which process is supposed to receive the data because port numbers act as identifiers for specific applications or services running on a device, allowing data to be directed to the correct service.

Correct answer: A

- 1 mark for marking correct answer
- 2 marks for giving correct reasoning for the correct answer.

Set C

Now that the team has tested you on basic OS concepts on virtualization, they want to check your understanding of basic networking. To this end, the team presents you with the following scenario: A process is executing in a machine A with IP address 192.156.13.21. This process is trying to communicate to another process running in machine B with IP address 192.156.13.29. The data contains confidential information. What component should be used for transportation and what should the addressing that the network stack in OS in machine A use to ensure that the data is delivered to machine B. Explain the reason **(3 points)**

A. Switch and port number
B. Switch, router and mac address
C. Hub and port number
D. Switch and mac address
E. Switch, router and port number

Notice that the two machines are on the same network (and subnet, but you are not expected to know that). This is evident from the fact that the first 3 octets are the same for both addresses. As discussed in class, intra-network communication is handled by **switches**. Routers are only needed for messages going outside the network or subnet.

Port numbers are an L4 (transport layer) feature and hence any answer including those is wrong.

Correct answer: D

- 1 mark for marking correct answer
- 2 marks for giving correct reasoning for the correct answer.

Q6

Set A

6. The team would like to understand from you the layer as per the OSI model which is going to ensure the service-to-service communication. Also, what protocol do you suggest to be used for ensuring that data is exchanged with low latency considering reliability and ability to control the flow of data. Explain the reason **(3 points)**
   A. Data link layer and TCP
   B. Transport layer and UDP
   C. Transport layer and TCP
   D. Network layer and IP
   E. Data Link layer and DNS

Service-to-service delivery is handled by the *Transport* layer.

Low latency is a plus point for *UDP*, but **reliability** and **flow control** are a clear marker of *TCP.* Both answers will be considered correct, as long as proper justification is provided.

Correct answer: B and/or C (A proper explanation should mention the tradeoffs and why you are choosing one over the other, or if both are chosen then what benefits do each have)

- 0.5 marks for marking the correct option
- 1 mark for explaining why Transport layer is chosen
- 1.5 marks for explaining why TCP/UDP is chosen.

Set B

6. The team wants your suggestion on which layer needs to be used to ensure the hop-to-hop communication. Moreover, they also want a suggestion from you on the protocol that can be used from a process perspective for ensuring that data is exchanged with low network overhead. Explain the reason **(3 points)**
    A. Data link layer and TCP
    B. Transport layer and TCP
    C. Data Link layer and UDP
    D. Network layer and IP
    E. Transport layer and UDP

Hop-to-hop delivery is handled by the data link layer. For low network overhead, UDP is the appropriate choice due to its being connectionless, lack of ordering, and lack of acknowledgements in addition to a smaller packet size.

Correct answer: C

- 0.5 mark for marking correct answer
- 1 marks for Data link layer explanation.
- 1.5 marks for UDP explanation.

Set C

6. The team would like to understand from the layer as per the OSI model which is going to ensure service-to-service communication. Also, what protocol do you suggest to be used for ensuring that data is exchanged at high speed with some acceptable loss? Explain the reason **(3 points)**
   A. Transport layer and TCP
   B. Network layer and TCP
   C. Transport layer and UDP
   D. Data Link layer and MAC
   E. Data link layer and UDP

Service-to-service delivery is handled by the *Transport* layer. UDP provides high speed connections with some losses due to its lack of acknowledgements, among other things.

Correct answer: C

- 0.5 mark for marking correct answer
- 1 marks for providing the correct reason for Transport layer.
- 1.5 marks for explaining UDP.