A

International Institute of Information Technology, Hyderabad

(Deemed to be University)

CS3.301 Operating Systems and Networks – Monsoon 2024

Quiz 1

Max. Time: 45 mins **Max. Marks: 20**

Roll No: _____ **Programme: _____**

Student's Signature: _____ **Invigilator's Signature:_____**

Special Instructions to the students

1. Answers written with pencils won't be considered for evaluation
2. Please **read the descriptions** of the questions (scenarios) carefully.
3. There are a total of six questions with four MCQs and carries 20 marks. For MCQs you can select one/all that applies as answers for a given MCQ. Please refrain from writing long explanations for MCQ questions. **Keep the explanations short and to the point (4-5 lines).**
4. **Feel free to use extra space** for any calculations/rough work but **they won't be considered for evaluations**.

Marks Table (To be filled by the Evaluator)

Question No / Marks	Initial	Final	Name of the Evaluator
1			
2			
3			
4			
5			
6			

General Instructions to the students

1. Place your Permanent / Temporary Student ID card on the desk during the examination for verification by the Invigilator.
2. Reading material such as books (unless open book exam) are not allowed inside the examination hall.
3. Borrowing writing material or calculators from other students in the examination hall is prohibited.
4. If any student is found indulging in malpractice or copying in the examination hall, the student will be given 'F' grade for the course and may be debarred from writing other examinations.

Best of Luck

Welcome to the Operating Systems and Networks Design Challenge. A team of system designers, OSNT, are teaching the concepts of OS and Networks through a design challenge. The challenge consists of a series of scenarios which they want the participants to work on. The goal of these challenges is in a way to test the existing knowledge of the candidate as well as a refresher for those who are familiar with OS and networks concepts. You have 45 minutes to solve the scenarios. Each scenario is awarded some points, and at the end of it, the final points you receive determine your position in the leaderboard.

1. As a first step the team wants to check your knowledge on process virtualization. You have been given an OS which is running on a single core CPU. Your first task is to figure out what happens to the process states and how they transition behind the scenes to support the virtualization. For simplicity, assume the following scenario:

   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.

   Based on the above scenario, the team wants you to explain the states of both the processes at the different instant of time with reasons. **(4 points)**

2. The team is building the shell. The team is planning to implement the below C code. They want this to serve as the base for the shell that they will be writing. However, they have some questions for you:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main (int argc, char** argv)
{
   printf("Id of the process is:  %d\n", getpid());
   int childId = fork();
   printf ("Returned Child ID value %d\n", childId);
   if (childId < 0)
   {
      fprintf(stderr, "The fork failed \n");
      exit(1);
   }
   else if (childId == 0)
   {
       printf(" I am the child process with id %d\n", getpid());
   }
   else
   {
       printf("This is the parent process of %d with process id: %d\n",childId, getpid());
   }
   return 0;
}
```

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

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)**

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

5. 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 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

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

Now that you have completed the task, it's time to wait to know your final points!!!

**********Extra Space*********

SET A
International Institute of Information Technology, Hyderabad
(Deemed to be University)
CS3.301 Operating Systems and Networks – Monsoon 2025
Quiz 1
Answer key

Ans 1
[1.5 points for correct answer]
(d) AI Tutor: 150, LangOS Main: 75 (each process maintains independent values)

[1.5 points for explanation]
Reason: After fork(), both processes have independent copies of variables. The child updates its own copy to 150, while the parent updates its own copy to 75. No sharing occurs.

Ans 2
[1.5 points for correct answer]
When a process running in kernel mode is preempted, the current “kernel stack pointer value” is saved into the process’s “PCB (process control block)”

[1.5 points for explanation]
Why: When preempted in kernel mode, the process is using its kernel stack. The OS must capture the CPU context—crucially the kernel SP (stack pointer) value—and store it in the PCB, so that on rescheduling the kernel can restore the registers and resume exactly where it left off. (The user stack isn’t what’s active here; the kernel stack is.)

Ans 3

Part A *[3 points]*

[1 points for shortest job first]

SJF (Gantt: P1[0–3] → P3[3–9] → P4[9–11] → P2[11–23])

- P1: Turnaround = 3−0 = 3; Response = 0−0 = 0; Waiting = 3−3 = 0
- P2: Turnaround = 23−1 = 22; Response = 11−1 = 10; Waiting = 22−12 = 10

- P3: Turnaround = 9−3 = 6; Response = 3−3 = 0; Waiting = 6−6 = 0
- P4: Turnaround = 11−4 = 7; Response = 9−4 = 5; Waiting = 7−2 = 5

Averages (SJF):
- Avg Turnaround = (3+22+6+7)/4 = 9.5
- Avg Response = (0+10+0+5)/4 = 3.75
- Avg Waiting = (0+10+0+5)/4 = 3.75

[1 point for short time completion first]

STCF (Gantt: P1[0–3] → P3[3–4] → P4[4–6] → P3[6–11] → P2[11–23])

- P1: Turnaround = 3−0 = 3; Response = 0−0 = 0; Waiting = 3−3 = 0
- P2: Turnaround = 23−1 = 22; Response = 11−1 = 10; Waiting = 22−12 = 10
- P3: Turnaround = 11−3 = 8; Response = 3−3 = 0; Waiting = 8−6 = 2
- P4: Turnaround = 6−4 = 2; Response = 4−4 = 0; Waiting = 2−2 = 0

Averages (STCF):
- Avg Turnaround = (3+22+8+2)/4 = 8.75
- Avg Response = (0+10+0+0)/4 = 2.5
- Avg Waiting = (0+10+2+0)/4 = 3.0

[1 point for round robin]

RR (q = 4, Gantt: P1[0–3] → P2[3–7] → P3[7–11] → P4[11–13] → P2[13–17] → P3[17–19] → P2[19–23])

- P1: Turnaround = 3−0 = 3; Response = 0−0 = 0; Waiting = 3−3 = 0
- P2: Turnaround = 23−1 = 22; Response = 3−1 = 2; Waiting = 22−12 = 10
- P3: Turnaround = 19−3 = 16; Response = 7−3 = 4; Waiting = 16−6 = 10
- P4: Turnaround = 13−4 = 9; Response = 11−4 = 7; Waiting = 9−2 = 7

Averages (RR):
- Avg Turnaround = (3+22+16+9)/4 = 12.5

- Avg Response = (0+2+4+7)/4 = 3.25
- Avg Waiting = (0+10+10+7)/4 = 6.75

Part b *[2 point]*

- Turnaround time (lower is better): STCF (avg 8.75) — smallest average.
- Response time (lower is better): STCF (avg 2.5) — best responsiveness for short/interactive jobs.
- Waiting time (lower is better): STCF (avg 3.0) — lowest average waiting.

Summary recommendation: STCF (preemptive SJF) performs best across all three metrics for this arrival pattern — it finishes short interactive jobs fastest and minimizes avg turnaround/response/waiting.

Ans 4
[1.5 points for correct answer]
Option (b) — 5ms at top, then 10ms/20ms below, with a moderate boost every 50ms — matches these goals.

[1.5 points for explanation]
- Give very small time slices at the top so interactive jobs get quick turnarounds and can finish before being demoted (reduces latency).
- Increase time slices at lower levels so CPU-bound jobs run longer once demoted (reduces context-switch overhead and improves throughput).
- Use a moderate priority-boost interval (not too frequent to waste CPU on boosting, not too infrequent to starve interactive tasks) so long jobs don't permanently starve short interactive tasks and responsiveness is preserved.

Ans 5
[1.5 points for correct answer]
b. UDP - includes checksum for error detection but has no flow control

[1.5 points for explanation]
Reason:
a. TCP (error detection, flow control, guaranteed delivery): Correct but adds latency via ACKs/retransmissions → not ideal for real-time.
b. UDP (checksum, no flow control): Correct and best for minimal delay, though reliability must be handled at app level.
c. TCP (connection-oriented, retransmission): True but retransmissions cause delay → poor for interactive typing.
d. UDP (auto
correction, flow control): Incorrect — UDP only detects errors, no correction/flow control.
e. TCP (faster due to lightweight nature): Incorrect — TCP is heavier than UDP, so slower for low-latency.

Ans 6

[1.5 points for correct answer]
Answer: (f) TCP

[1.5 points for explanation]
a. FTP: Incorrect — used for file transfer, not email.
b. SMTP: Incorrect here — SMTP is an application-layer protocol for sending mail, not a transport protocol.
c. HTTP: Incorrect at transport layer — its application-layer (though modern email clients may use HTTPS for APIs).
d. RTP: Incorrect — used for real-time audio/video streaming, not email.
e. UDP: Not suitable — no reliability, ordering, or guaranteed delivery → bad for email.
f. TCP: Correct — provides reliable, ordered, connection-oriented delivery, essential for email protocols (SMTP, IMAP, POP3) to function correctly.

BONUS

Question 7 ***(3 points)***

• Q1. Virtual runtime update equation *(1 point)*

– Correct formula: $\Delta vruntime = \Delta t_{exec} \times 1024 / w_{task}$
(1.0 point)

– Partial credit: correct proportional reasoning without full formula
(0.5 points)

• Q2. Effect of nice values *(2 points total)*

– (a) Rate of increase of vruntime *(1 point)*

∗ Low nice (high priority): Higher weight ⇒ slower vruntime increase *(0.5 points)*
∗ High nice (low priority): Lower weight ⇒ faster vruntime increase *(0.5 points)*

– (b) Amount of CPU time received *(1 point)*

∗ Low nice (high priority): Receives more CPU time (*0.5 points)*
∗ High nice (low priority): Receives less CPU time *(0.5 points)*

Total: 3 points

Question 8 ***(2 points)***

• Part (a): Terminal Output (1 point)

– Award *1 point* if the terminal output is exactly:

BEFORE

– Award *0 points* otherwise.

• Part (b): File Content *(1 point)*

– Correct final contents of output.txt:

CHILD
EXEC

PARENT

(1.0 point)

– Partial credit: Identifies redirection into output.txt correctly,
but with missing/misordered lines *(0.25 points)*

Total: 2 points