International Institute of Information Technology, Hyderabad

(Deemed to be University)

CS3.301 Operating Systems and Networks

End Semester Examination

Max. Time: 3 Hr **Max. Marks: 75**

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 eleven questions with five MCQs and carries total of 75 marks. For MCQs you can select one or more options (if necessary). Please refrain from writing long explanations for MCQ questions. **Keep the explanations short and to the point.**
4. Please **state any assumptions** made clearly.
5. Use pseudocode when necessary (you are not required to provide the exact C implementation)

Good Luck

Congrats on being selected as a consultant for the IndixOS design team!

IndixOS is a fictitious Unix-based Operating system that is being designed in India by a team of 20 system designers. IndixOS is also expected to have a network stack implementation as per the OSI model. In order to make the development easier, the IndixOS team has been split into five sub-teams of 4 members each, namely: Fork team, responsible for working on the process virtualization aspects; Mem team, for developing memory virtualization; the Sem team, working on the concurrency related aspects, F&D team working on the persistence aspects and Net team working on the networking aspects. The teams aim to put efficiency (time, memory usage, energy, etc.) and fault tolerance as the primary goals for building the IndixOS.

Recognizing that poor design practices are a common issue resulting in long-term maintenance challenges, the overseeing board of IndixOS has hired you as an expert consultant system designer to review their existing design and provide guidance to the teams regarding the design of the overall IndixOS in general as well as certain specific aspects.

As the consultant, you are given 180 minutes to complete several tasks. You must complete these tasks within the allocated timeframe, as your services have been billed accordingly. Each task is assigned a specific number of points, and the cumulative score of all the tasks will determine your final rewards (based on total points), with a maximum of 75 points.

1. The *fork team* have created two APIs for supporting process virtualization: i) repl(), which is similar to fork(); ii) execute(), which is the same as the exec() and iii) wait() as offered by the standard Unix OS. The team has some questions related to the implementation, which requires your input:
    a. The team has developed a kernel and a shell on top of it. However, they want to implement support for a new shell command, "fwc" which takes a file name as input and gives out the number of words in the file as output (eg: *fwc file.txt* gives "20" as output indicating 20 words). Since shell itself is an executing process, they want your suggestion on implementing this command. Please explain how the existing system calls can be used to achieve this. **(3 points)**

b. The *fork team* learned from the *Mem Team* that they might be using Paging for Memory management. Assuming that it will be paging, Team fork wants your input on how the memory management will work when performing calls like repl() and execute() **(3 points)**

c. At any given point, there will be many processes executing in the IndixOS. At present, the team relies on the cooperative approach to pass control back to the OS. However, the team is unclear about how a pre-emptive approach can be implemented to get control back to the OS during process execution. For instance, assume that process B is getting executed, and if it makes some system call during execution, then as per their current implementation the OS will be able to get control and execute some other process C. Apart from this, the team requires the OS to have a better mechanism which allows the OS to get control *as and when required*. The team requires your suggestion based on your understanding of Unix about the mechanism and protocol that the OS needs to use to ensure that it can seamlessly switch between multiple processes. **(4 points)**

2. *The Mem Team* has considered multiple policies for memory virtualization, and finally, they have decided to use Paging. However, they are not sure about certain aspects and require your suggestions:
   a. The team is thinking of dividing the virtual memory into pages of different sizes and then mapping them to physical memory, where, again, it is divided into blocks of different sizes to support different processes that may have different memory requirements. What are your observations and comments on this approach? **(3 points)**

   b. The team is planning to use a FIFO approach for page replacement where the system has a cache size of 3. The team claims that FIFO is the best policy for page replacement by considering an example trace: 0, 1, 2, 0, 1, 3, 0, 3, 1, 2, 1. The team would like to know your opinion**. (3 points)**

   c. The team is considering a 32-bit address space with 2 KB pages. The team estimates each entry in the page table to consume 4 bytes. On average, the *Mem team* estimates any machine to support around 100 processes. This results in a memory overhead. Moreover, for any address translation, the page table needs to be first accessed, and then the page table entry needs to be fetched. This results in two accesses to the memory for each address translation. Is there some better mechanism that you can suggest to the team to handle these issues? Please elaborate on the mechanism. **(4 points)**

3. The *sem team* responsible for managing concurrency is testing out applications to test the concurrency aspects. They have implemented the notion of semaphore that is made available through the POSIX library. The team plans to test this by creating a simple application similar to Google Docs. However, they want you to test their semaphore implementation. The constraint they have for the application is that multiple threads should be able to read a document at the same time, but only one writer should be able to write at any given instant of

time. Moreover, they also want to ensure that there is no starvation of thread that is waiting to write to the document. Can you devise a solution to address these constraints (Pseudocode)? (Clearly state the assumptions made and reason on the solution) **(10 points)**

4. The *F&D Team* is developing a simple file system module for the IndixOS. The team is now developing a small proof of concept with a disk that has eight sectors, each comprising eight blocks of size 4 KB. The team wants to support storage of different files and directories with the given disk space (as well as their metadata). The F&D team wants the disk to support storage and retrieval of 224KB of data. They seek your opinion on how certain aspects of the file system can be implemented:
    a. According to you, what is the maximum number of files/directories that can be created on the disk based on the constraints (assuming that the metadata of the file or directory consumes 256 bytes), and what does this imply? **(3 points)**

    b. The team wants to implement a read access operation to "/user1/file.txt" (e.g., cat /user1/file.txt). They would like your opinion on handling the read access in this simple file system. Further, how many reads and writes on the disks will be performed when performing this read operation? **(5 points)**

    c. Given that the file system may issue an operation to read from or write to blocks of disks, what are some mechanisms that the file system, in tandem with the disk controller, can use to improve the efficiency of the reads and writes **(2 points)**

5. The Net team has developed a simple web application to test their TCP implementation. The web application is a simple process that will be sending a stream of data 11100110011001101101010101010101 to another process in a different machine (running IndixOS). Moreover, the team also would like to develop some mechanisms to guarantee reliability during the transfer. However, they are still determining how this can be implemented, and they require your suggestions:
    a. The Net Team would first like to understand what kind of APIs the OS needs to provide to make this communication happen (based on Unix implementation) and what is the role of the OS when such communication happens. **(3 points)**

    b. How can the process in the destination machine validate the incoming data? **(4 points)**

    c. Further, how can the Net team provide guarantees on data reliability and flow control? **(3 points)**

6. The *Net Team* has completed the network stack development following the OSI model. Now, they want to perform some tests to check if everything is working fine. They have a machine with an IP address 192.168.1.25, MAC 9B:8C:6D:5E:3E:2F, with a subnet mask of /24. The public IP address of the network is 17.84.18.65, and the IP address of the router is 172.18.12.92; MAC address 1D: 7G:98:1C:1A:2D. The team wants to understand how the data transfer can be achieved in the following cases:
    a. The destination machine has an IP address of 192.168.1.32 with a subnet mask 255.255.255.0, MAC 8A:9B:1C:2F:9F:6D. From L2 perspective, can you help the team understand how this communication can work with reasoning **(5 points)**

b. The destination machine has an IP address of 10.8.17.65 with a subnet mask of 255.255.255.0 and a MAC address of 1A:2B:3C:4D:9C:5A. The machine is in a network with an outgoing IP address of the router being 18.72.19.85 and the internal address of the router is 10.8.17.89 /24. Can you explain to the team how the data transfer works in this scenario? (*Hint:* Think about both L2 and L3) **(5 points)**

7. *The Net Team*, in order to test their network stack, has decided to connect a machine with IndixOS to a network (172.18.20.x/24). However, the team wanted to check with you how the host machine can get an IP address in the given network. What can be the protocol that can be leveraged, and how does it work? **(3 points)**
    a. DNS
    b. HTTP
    c. DHCP
    d. ARP

8. The F&D team wants IndixOS to have support for RAIDs as the OS will primarily be used in the data centre machines. The team is expecting the system to have workloads with more random reads and writes as opposed to sequential reads and writes. Which of the following RAID levels do you recommend and why? **(3 points)**
    a. RAID 0
    b. RAID 1
    c. RAID 4
    d. RAID 5

9. The *sem team* wants to test their implementation with an I/O intensive operation where they want to create a program that can process a file consisting of 100000 records. In order to make it faster and to test the efficiency of their implementation, the thread team are planning to write a program which will use 10 threads where each thread reads 10000 records and processes them independently, so in total, all the 100000 records will be processed. However, the team is not sure which of these is required to complete the implementation. Which of these do you recommend and why? **(3 points)**
    a. Make use of locks. No need for condition variables
    b. Only condition variables are needed. No need for locks
    c. Both locks and condition variables are needed.
    d. No need for either locks or condition variables.

10. *The Fork team* wants to check how IndixOS can seamlessly switch between different running processes at any given point. The team expects IndixOS to have a mix of workloads with a lot of I/O-intensive workloads and some CPU-intensive workloads. Assuming that, as in any real case, there will not be any information on the completion time of each process and any process can arrive at any time, the team wants to check with you which could be a better policy to use for scheduling. Which one of the following scheduling policies do you recommend and why? **(3 points)**
    a. Multi-level feedback queues

b. First come, first serve
c. Round Robin
d. Shortest Completion time first

11. The *Net team* has built the application layer support in their OS network stack by providing support mainly for HTTP/1.1. Before opening IndixOS to the public, the team wanted to do a final test where multiple systems using IndixOS are connected to an extensive network. One machine was designated as the server, and other machines as the clients. The team observed that since many client machines were sending HTTP requests to the server, the server was not able to handle the load, and it crashed. What do you think is the quick solution that the team can implement to proceed with the test, and how? (**3 points)**
    a. The team can make use of cookies.
    b. Web caches can be leveraged.
    c. The team should use HTTP/1.0 instead.
    d. Make use of proxies.

**********Now it's time to wait for final points**********