Final Exam

Due Dec 11 at 11:59pm **Points** 100 **Questions** 10 **Time Limit** None **Allowed Attempts** Unlimited

Instructions

Please answer the following open-ended questions IN YOUR OWN WORDS and support your answers with descriptive examples. You will not receive credit for generic definitions or explanations. I want to over-emphasize that you should focus your effort on describing and using examples to justify and support your responses.

Good luck!

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	92 minutes	0 out of 100 *

* Some questions not yet graded

Correct answers are hidden.

Score for this attempt: **0** out of 100 *
Submitted Dec 6 at 9:32pm
This attempt took 92 minutes.

Question 1

Not yet graded / 10 pts

What is an operating system? What is the purpose of an operating system? What are the different types of operating systems?

Your Answer:

What is an OS:
-An operating system is system software that manages computer hardware, software resources, and provides common services for computer programs.
Purpose of an OS:
- Manage the computer's resources, such as the central processing unit, memory, disk drives, and printers.
- Establish a user interface.
- Execute and provide services for applications software.
OS type:
- Batch Operating System
- Time-Sharing Operating Systems
- Distributed Operating System
- Network Operating System
- Real-Time Operating System

Question 2

Not yet graded / 10 pts

From an architectural design perspective, how does the Operating System protect itself from malicious applications? Provide one or more examples to justify your answers.

Your Answer:

OS separate 2 parts: kernel space and user space, kernel to protect itself from security breaches, such as runaway processes, memory-access violations, ...
only kernel space can access hardware device. So in user mode, it has limited capability to execute.

Question 3 **Not yet graded / 10 pts**

What is the difference between a process and a task? Be sure to explain the resources available to each and any shared resources between other processes and/or threads. Provide one or more examples to justify your answers.

Your Answer:

Process is upper level, task is at lower level. Process does not involve input/output operations while A task includes I/O operations.
processes can use IPC(shared memory, message queue, socker, pipe...) to access the common resources/memory.
Threads have common static, heap memory while the processes memory are different.
Example

```
int a;
void *myThreadFun(void *vargp)
{
    sleep(1);
    printf("Printing GeeksQuiz from Thread \n");
    return NULL;
}
int main()
{
    int a;
    pthread_t thread_id;
    pthread_create(&thread_id, NULL, myThreadFun, NULL);
    pthread_join(thread_id, NULL);
    fork(); // create child process
    return 0;
}
```

variable a is not the same memory in parent and child process but it's same in parent/child process and its created thread.

Question 4 **Not yet graded / 10 pts**

Describe each of the process states using an example of an incoming process.

Your Answer:

The list of process states:

- New: The OS will create a process identifier and structure for process management such as memory table, file table, process control block when an incoming process come into OS. This called New state

- Running: The CPU now serve this process, it mean that the CPU execute it's instructions.

- Waiting: The process waiting for some events to occur such as I/O, ... It will bring to waiting queue. It's state is called Waiting.

- Ready: When process created and ready for execute, it need to wait for assign to a processor. The processor may busing for other process. This called Ready state.

- Terminated: when process done his job or finished execution.

Question 5 **Not yet graded / 10 pts**

From a program/application perspective, what is the purpose of plumbing and piping (e.g. hint: related to your implementation of the shell).

Your Answer:

The piping and plumbing are used to make the output of one program is sent as input to another program.
they are useful when you need data from one program and doesn't wanna retype all the output.

Question 6

Not yet graded / 10 pts

What is preemption? Describe the difference between a preemptive and a non-preemptive scheduler. Support your answer using examples of specific preemptive and non-preemptive scheduling algorithms.

Your Answer:

What is preemption:

the preemption is the action of temporarily interrupting a process or task that is being executed without requiring its cooperation.
The process or task will resume execution at later time.

Preemptive and non-preemptive:

Preemptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process. The example of scheduling algorithms is Round Robin. Each process will be execute in special amount time, and after that amount time, if it not done yet, the CPU will stop it and next process will execute next.

Non-preemptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst. The example of scheduling algorithms is First Come First Service. When a process being execute CPU it will keep execute until done it's job. It cannot stop or interrupt while it executing.

Question 7

Not yet graded / 10 pts

What is the purpose of the short, medium, and long term schedulers? Provide one or more examples to justify your answers.

Your Answer:

Long term scheduler can regulate the program and select process from the queue and loads them into memory for execution.
Medium-term scheduling notifies to handle the swapped out-processes. The running process can be paused if it has an I/O request.
Short term scheduling is to boost the system performance according to set criteria.

Example: When you start a program that needs data from out of main memory. Swapping programs in and out of memory is the job of mid-term scheduler.

Question 8

Not yet graded / 10 pts

What are the performance metrics of a scheduler and how are they calculated? Use examples to justify your answer.

Your Answer:

The performance metrics of a scheduler are:

- Utilization: We need maximum the utilization. To calculate utilization, we need calculate the ratio time between busy time and total time. Example: for CPU scheduler, we need maximum CPU Utilization, it mean that keep CPU busy as much it can. The utilization is percent of CPU busy.

- Throughput: we need maximum the throughput. To calculate throughput we need calculate number of tasks in unit time. Example: throughput of CPU scheduler is execute as much process in unit time.

- Response time: we need minimum response time. To calculate response time, we need calculate the time the process or task waiting to be execute. Example: when CPU scheduler scheduling for list of the processes, the time of each process in waiting queue is response time.

- Waiting time: we need minimum waiting time. To calculate waiting time, we need calculate the the total time spent by the process in the ready state waiting for CPU. Example: when CPU scheduler scheduling for list of the processes, the time of each process is spent by the process in the ready state waiting for CPU (the process may interrupted during executing put to waiting state).

Question 9 **Not yet graded / 10 pts**

Describe the synchronization tools we studied using examples to support your understanding of how they work.

Your Answer:

The synchronization tools:

- We have many processes and they will shared a memory. Example, the Printer.

Question 10 **Not yet graded / 10 pts**

Describe how data transfers over connected sockets (Send() And Receive()) from a server and multiple clients perspective? What are the conditions in which a server can support multiple incoming connections and requests?

Your Answer:

When server sends data to multiple clients (must use TCP), the data from user space to kernel space. In kernel space, data will be transfer to all clients that bind through port that set before(not routing if use UNIX domain socket).
Every connection socket has a different memory to handle in server in kernel space.
And on the side of the clients, use receive to read data from port of the bind socket. Data is sent to port and transfer from kernel to user space.

Conditions to server can support multiple incoming connections and request:
- RAM must be enough to handle request and create a connection socket.
- Not reach to the maximum clients that server set must use TCP protocol.

Quiz Score: **0** out of 100