Operating Systems, Evaluation 1

CS30002, Spring 2020

9:00 am to 10:00 pm, 19th February, 2020

Full marks: 40

Answer ALL questions

IMPORTANT INSTRUCTIONS

Taking the exam: You need to log into Google meet (personalized links will be provided close to the exam date), keep your video on during taking the test (so that we can monitor you during the exam). YOU HAVE TO USE PEN AND PAPER TO GIVE THE EXAM.

Decorum: Throughout the examination, you are strictly expected to have their cameras on, directing towards their workspace including themselves. Arrange your laptops/desktops/mobiles beforehand to save time during the examination. Disconnecting video for a long duration will be grounds for suspecting malpractice.

You need to keep your workplace and your hands visible to us. However, avoid the visibility of your answers to the rest of students. Once you open your question paper, refrain yourself from using your PC/laptop/mobile for searching for anything or typing during the exam.

Submission: You can do either of two things -- (i) take pictures of your answer script pages, name the pictures page1.jpg, page2.jpg, page3.jpg, etc., zip the pictures and upload the zipped file via CSE Moodle, or (ii) Put all the pages sequentially in a pdf file and upload the pdf to CSE Moodle. Note that the scheduled time includes both writing time and time needed to upload your answer-script.

Submissions must be through the course Moodle, by 10:00 AM (according to the Moodle clock). If you miss this deadline for Moodle submission, then you need to email your submission to TA Anju Punuru (anjupunuru@gmail.com) within 10:10 AM; there will be a penalty of 10 marks for such late submissions. No submission will be allowed after 10:10 AM. If any submission reaches Anju's mailbox after 10:10 AM (according to the timestamp of the mail), it will be rejected.

Tip: Install Adobe scan on your mobile phone to make the whole process easier. In that case, your laptop acts as a camera, while you are using your mobile for checking the questions, scanning and uploading the answers. [This is only a suggestion that can make things simpler for you, not a necessity.]

Policies: Note that, if we face problems with your answer script, e.g., cannot open your submitted zipped file, cannot read the text in pictures (due to bad resolution), cannot determine the page order from the file names (or if the pages in the pdf are jumbled up), it will affect your marks.

Malpractice: If any group of students is found to have similar answers/working in their answer sheets, ALL of them will receive the maximum penalty with no grace. We will not distinguish between who supplied answers and who copied; everyone involved will receive the maximum penalty. We expect you to NOT take help from the internet, your copies, textbooks, slides or video recordings during the exam. Note that this is NOT an open-book exam. Also, you should NOT discuss answers with anyone during the scheduled exam time. If found otherwise, you will be penalized.

PLESE WRITE YOUR NAME AND ROLL NO. ON THE TOP OF THE FIRST PAGE OF YOUR ANSWER SCRIPT. WE WILL NOT EVALUATE YOUR ANSWER SCRIPT WITHOUT IT.

Question 1: State with clear justifications whether the following statements are true or false. *No marks will be awarded if the justification is not correct*. The Justification should be between 1-3 sentences (marks might be deducted for longer justifications). **[8 × 2 = 16]**

1.1. Operation of a time-sharing system is identical to operation of a traditional multiprogramming system executing the same programs if the time slice exceeds the CPU burst of every program.

1.2. The instruction to change the processor from user mode to supervisor mode is a privileged instruction.

1.3. For multithreaded programming, we need to create shared memory segments through which the threads can access common data.

1.4. Pre-emptive CPU scheduling algorithms can result in shorter average waiting time as compared to non-pre-emptive scheduling algorithms.

1.5. The **`signal(pid,sig)`** function call can be used by a user-level process to send a designated signal to some other process.

1.6. Local variables defined within a function cannot be used as shared data between a parent process and a child process, but can be used as a shared data between two execution threads of a process.

1.7. It is possible to have same values printed on the screen by both parent and child process after execution of the following code:

```
int a;
fork();
printf("Address of a: %p\n", &a);
```

1.8. The message “Hello” will be printed 64 times by the following C code segment.

```
pid = fork();
if (pid == 0) { fork(); fork(); fork(); }
else { fork(); fork(); }
printf (“\n Hello”);
```

Question 2: Answer the following. **[2 + 2 + 3 +2 = 9]**

2.1. Why is it necessary to enter a kernel via a system call as opposed to a normal function calls (like you wrote in your PDS assignments)? State two distinct reasons (1-2 sentences for each reason).

2.2. Can an unnamed pipe in Unix created using the pipe() system call be used to communicate between any two processes in the system? Justify your answer (1-2 sentences for justification)

2.3. Recall that in a process control block (PCB) contain the file descriptors for files opened by that process. Imagine a process opens the file “file.txt” and read a line, then it calls fork() to create a child process.

(i) Can the child process read from “file.txt” immediately after fork()? Justify your answer based on how fork impacts the PCB.

(ii) Will the child process’s access (or non-access) to “file.txt” depends on what the parent process is doing (i.e., parent reopening/closing the file)? Justify your answer based on how fork impacts the PCB.

2.4. In an actual implementation of the operating system, how exactly are the following process state transitions take place (1-3 sentences each):

i. Running to Ready
ii. Waiting to Ready?

Question 3. Answer the following: **[5 + 5 + 5 = 15]**

3.1. Consider three processes, all arriving at time zero, with total execution times of 10, 20 and 30 units, respectively. Each process spends the first 20% of execution time doing I/O (reads the input data), the next 70% of time doing computation, and the last 10% of time doing I/O again (prints the results). The operating system uses a ***shortest remaining compute time first*** scheduling algorithm and schedules a new process either when the running process gets blocked on I/O or when the running process finishes its compute burst. Assume that *all I/O operations can be overlapped as much as possible*. For what percentage of time does the CPU remain idle?

3.2. Consider a system with 100 processes in the Ready Queue. If round robin scheduling algorithm is used with a scheduling overhead of 5 microseconds, what must be the time quantum to ensure that each process is guaranteed to get its turn at the CPU after 1 second?

3.3. Consider the following CPU processes with arrival times (in milliseconds) and length of CPU bursts (in milliseconds) except for process P4 as given below:

Process	**P1**	**P2**	**P3**	**P4**
Arrival Time	**0**	**1**	**3**	**4**
CPU Burst Time	**5**	**1**	**3**	**Z**

If the average waiting time across all processes is 2 milliseconds and pre-emptive shortest remaining time first scheduling algorithm is used to schedule the processes, what will be the value of **Z**?