CS330: Operating Systems
Quiz#2

Name:
Roll No.:

1. Consider the following description of five processes P_0, P_1, P_2, P_3, and P_4.

- P_0: total duration is ten seconds; the first five seconds are spent in an I/O burst and the remaining five seconds are spent in a CPU burst.
- P_1: total duration is ten seconds; the first five seconds are spent in a CPU burst and the remaining five seconds are spent in an I/O burst.
- P_2: total duration is ten seconds; the first five seconds are spent in a CPU burst, the next three seconds are spent in an I/O burst, and the remaining two seconds are spent in a CPU burst.
- P_3: total duration is ten seconds; the entire duration is spent in one CPU burst.
- P_4: total duration is ten seconds; the entire duration is spent in one I/O burst.

Which of these processes are not realistic and cannot be found in a real-world computer system? **(2 points)**

Solution: Every process must start with a CPU burst and end with a CPU burst. This is because the first instruction that the process executes will be part of a CPU burst and the `exit` system call executed at the end is part of a CPU burst. So, processes P_0, P_1, and P_4 are not legitimate processes and cannot be seen in real-world.

Grading policy: Two points for mentioning the correct set of three processes. No partial marks because any illegitimate process not mentioned in the answer shows lack of basic understanding about processes.

2. A uniprocessor computer system uses a short-term process scheduler that allows the currently scheduled process to run until it encounters an I/O burst, at which point a context switch is executed and a different ready process is scheduled to run. Suppose two processes A and B are submitted to this system at the same time and A is scheduled first. The processes are specified below.

- A: total duration is twelve seconds; the first five seconds are spent in a CPU burst, the next four seconds are spent in an I/O burst, and the remaining three seconds are spent in a CPU burst.
- B: total duration is $(\tau_1 + \tau_2 + \tau_3)$ seconds; the first τ_1 seconds are spent in a CPU burst, the next τ_2 seconds are spent in an I/O burst, and the remaining τ_3 seconds are spent in a CPU burst.

Assuming that there are no other processes in the system, write down the minimum value of τ_1 and maximum value of τ_2 such that the CPU idle time is minimized. Ignore all context switch overhead. **(3 points)**

Solution: The four seconds of I/O burst in A needs to be covered by the first CPU burst of B. **So, τ_1 should be at least four seconds.** Next, the I/O burst of B needs to be covered by the second CPU burst of A. **So, τ_2 cannot be more than three seconds.** These values of τ_1 and τ_2 achieve zero CPU idle time and that is the minimum achievable.

Grading policy: 1.5 points each for τ_1 and τ_2.

3.(a) A program has ten processes (including the parent process which creates nine children). Each process computes the median of a distinct array and communicates the result to the other nine processes by using UNIX message passing. Assume that there is a message queue between every pair of processes. The message queues are created by the parent process before creating the children using `fork`. Each message queue has a distinct key and these keys are pre-defined. Compute the total number of system calls arising from creation of the message queues, sending the messages, receiving the messages, and deleting the message queues. **(2 points)**

Solution: Number of queues = 45. Parent makes 45 `msgget` calls before `fork` to create these queues. Every process makes nine `msgsnd` and nine `msgrcv` calls. Finally, one `msgctl` call is needed for each queue. Therefore, the total number of system calls = 45 + 180 + 45 = 270. If you have assumed that each child makes nine `msgget` calls to get the queue descriptors of the nine communication queues, that would make the total number of system calls 351. I will accept both solutions, although it is important to note that the additional `msgget` calls in the second solution are unnecessary and introduce inefficiency. The `fork` call will automatically copy all the queue descriptors to the child space.

Grading policy: Two points for correct answer. No partial marks.

3.(b) Now consider that the problem of 3(a) is solved using shared memory. A single shared memory region is created holding an array named `median` having ten elements. Process i deposits its result in `median[i]` for $i \in [0, 9]$. Compute the total number of system calls arising from setting up the shared memory region, the communication, and the removal of the shared memory region. **(2 points)**

Solution: One `shmget` call, one `shmat` call per process, one `shmdt` call per process, and one `shmctl` from parent. Total number of system calls = 22. Here also if you have assumed one `shmget` call per child, the answer would be 31. I will accept both answers.

Grading policy: Two points for correct answer. No partial marks.

4. Consider the following C code segment where `f` is a function not shown. Write down the output that is printed. Assume that `x` and `y` are private variables and do not have global scope. **(1 point)**

```
int x = 2;
void *child_stack = malloc(16384);
int y = clone(f, child_stack+16384, CLONE_VM | CLONE_THREAD | CLONE_SIGHAND, NULL);
if (y == getpid()) x++;
else x--;
printf("%d\n", x);
```

Solution: The return value of `clone` in this case is the thread id of the created thread. This will not match the pid of the parent even though the child would have the same pid. **So, the output is 1.**

Grading policy: One point for correct answer. No partial marks.