Roll Num: MS17BTECH11013 **Name: P. Pavan Kalyan**

CS3510: Operating System 1
Quiz 1

---

1. Programs in user mode have no direct access to memory and hardware resource. Programs call system calls and they are executed in the kernel-mode by the OS. Here the mode is switched from user mode to kernel mode and runs those special instructions on behalf on user mode. Here the OS ensures the safety by putting some memory protections so that they can't be accessed and modified. In this way the OS puts some restrictions on important memory locations so that they can't be modified when program run in user mode and call system calls. The OS simply throws an error saying that it can't be accessed. Similarly, we can see in a C program that when we try to access a value out of bound of an array it simply shows segmentation fault. Here the program tried to access the other memory locations in user mode but the OS recognizes these errors and throws errors.

2. In this symmetric multiprocessing system as shown in the figure (in Txt book).

There are two process processor0 and processor1 which are having their own CPU, registers and cache. When processor0 fetch a data such as var = 0 which is in the main memory it stores it in the cache similarly the processor fetch the same var = 0 which is in the main memory it stores it in own Cache. When processor0 tries to modify the value in 'var' it only changes it in the processor0 cache but they are not accessed by the other processor. Similarly, when processor1 changes value in 'var' to 5 it only gets modified in the cache of processor1. Here the individual cache is local(private) to the processor only. So, data residing in memory could in fact have a different value in each of the local caches.

3. The circumstances under which the line of code marked printf("LINE J") will be executed is when there is an error in executing execlp in the child process (pid = 0). Because we know that execlp replaces rest part of the code with it's instruction. When a program calls fork, child process is created and with pid as 0 and both parent and child process will be running. In child process instruction if pid == 0 becomes true and then execlp will be executed if there are no errors then child process finishes because execlp replaces rest of the code but if it's unsuccessful(errors) then the instructions after it will be executed.

4. output at lines X will be

(Assumption: I kept a '\n' in the printf statements for ease of understanding) TA also approved

```
CHILD: 0 1
CHILD: 1 0
CHILD: 2 -1
CHILD: 3 -2
CHILD: 4 -3
```

Output at lines Y will be

```
PARENT: 0 0
PARENT: 1 -1
PARENT: 2 -2
PARENT: 3 -3
PARENT: 4 -4
```

Explanation: Before we execute the fork() command there is an array nums which is initialized to 1 for all values. nums[i] = { 1, 1, 1, 1, 1,}.

When we call fork there are two process running now parent and child both will have array nums in them with the values initialized before. In the child process pid == 0 becomes true and we modify the nums array in it and print the values. But these changes are also in the child process. So nums become Nums[i] = { 1, 0 , -1, -2, -3} and then we print it.

In the parent process nums will have the values nums[i] = { 1, 1, 1, 1, 1,} even though we wait for the child process to finish executing because the changes done to the nums in the child process are only applied to it because it has it's own memory space allocated to it initially with the same variables and array in the parent process when the child process terminates these changes are not applied to the parent process nums array. And then we do the operation of multiplying -i value to each value in the nums here nums will have value { 1, 1, 1, 1, 1,} after doing the operation nums will have the value { 0, -1, -2, -3, -4} as i value starts from 0 and ends till 4.

5.

A)

Synchronous communication

1. It blocks other operation till the current operation Completes.

Asynchronous Communication

1. It simply initiates the new operation replaces the old operation.

B)

Fixed-sized messaging: Here the buffer size is fixed means only messages of size less than or equal to the size of buffer can be send. The disadvantage is that buffer size is fixed.

Variable-sized messages: Here there is no restriction on the message size. Messages of any length can be sent.