Home Assignment 1

Subject: Operating Systems

Dhruvi Pandya

121011

1. Usually, Linux schedules the parent and child processes independently; there’s no guarantee of which one will run first, or how long it will run before Linux interrupts it and lets the

other process (or some other process on the system) run.So, as soon as fork() is encountered, the kernel creates the copy of the parent process with different ID and address space. After this, the next instruction after fork is executed first. And then onwards, both the processes get executed independent of each other. But, here assuming that child process gets executed first, the out put will be like:

Child sees i=0

Parent sees i=0

Child sees i=1

Parent sees i=1

Child sees i=2

Parent sees i=2 ….............OR.......

Child sees i=0

Child sees i=1

Child sees i=2

Parent sees i=0

Parent sees i=1

Parent sees i=2 …............OR.......

Child sees i=0

Child sees i=1

Parent sees i=0

...

So, after the first occurrence of the child process, the order between the Child and the Parent process can be random but the i's will be in ascending order.

1. In Threading,unlike fork(), the threads created does not have separate address space and the data; they share every asset of its parent process. Also, when we say a process is being executed, we mean that its threads are carrying out the different tasks of the same parent i.e. the threads are being executed. So, the output will look like:

Parent says a: 1

id: 2 a: 2 b: 1

id: 1 a: 3 b: 1

^C

dhruvi@ubuntu:~/Desktop/ULScheduling/Assignments/sub\_1$

3. m>>n : Number of System calls decrease. Overhead decreases. Therefore, very much favourable.

m>n : Reasonably good.

m=n : one user process per one kernel process. Therefore, it's not multi-threading.

m<n : More overhead.

m<<n : Number of System calls increase. Overhead increases. Therefore, should be avoided.