**Himanshu Ojha**

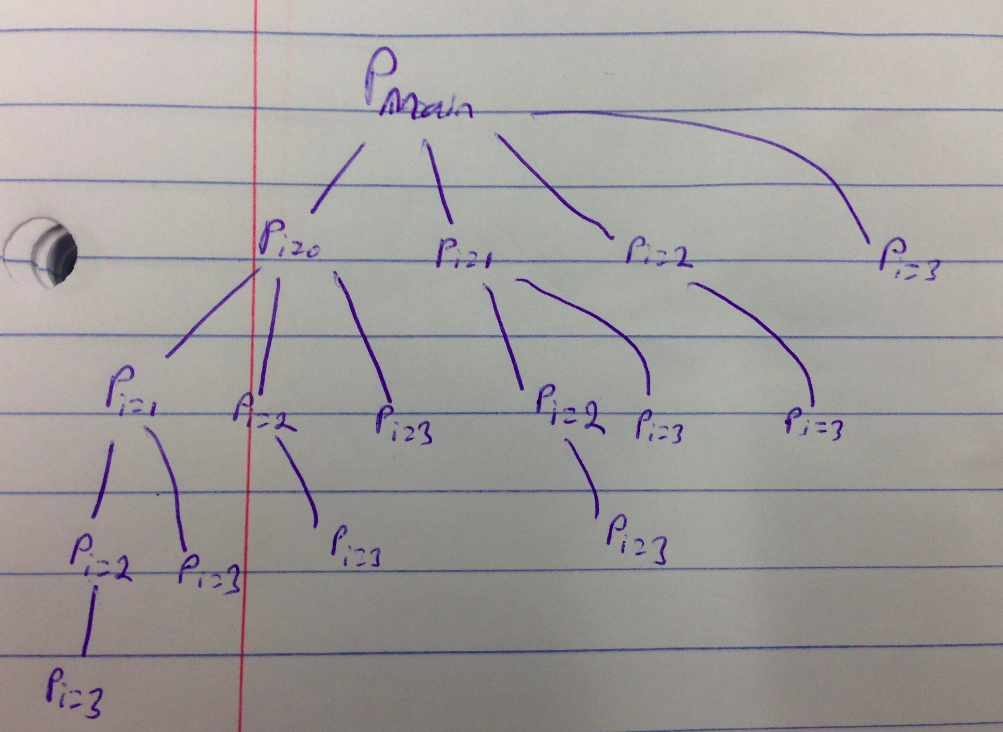
**CS4414 (F2016) Operating Systems**

**Homework 1**

**Spring, 2016**

**I have neither given nor received unauthorized information on this assignment.**

(1)



*16 processes are created.*

*The notation of the image is that P means process. P\_{i=0} means the process created when i=0. Thus what happens is that the main process creates 4 children. The first child creates 3 children. And so on.*

*The main process creates 4 children. The first child is created when i=0 for the main process. When the code for this first child runs, it starts at i=0. Thus the first fork() that is run for this first child only returns 0 in order to indicate the current process is a child. This 'recursively' occurs for each successive process until each process hierarchy ends at i=3. This ends up creating a total of 16 processes.*

(2) *The line marked printf(“LINE J”); is called in the child process when the execlp function fails. The execlp function fails when /bin/ls file does not exist in either PATH or in the current directory. The execlp function returns a non-zero number when it fails.*

*Execlp function replaces the current process image with the new processes image when the function succeeds. In doing so, the printf(“LINE J”); line is overwritten. Thus when the function succeeds, the line is not run.*

(3)

*A: 0*

*B: 2603*

*C: 2603*

*D: 2600*

*For the parent process, pid=2603 because pid represents the pid of the child.*

*For the parent process, pid1=2600 because pid1 represents the pid of the currently running process which is the parent process.*

*For the child process, pid=0 because pid is set to be 0 for the child process. This is just how fork() works.*

*For the child process, pid1=2603 because pid represents the pid of the child.*

(4)

*The output will be:*

*CHILD 0 CHILD -1 CHILD -4 CHILD -9 CHILD -16 PARENT 0 PARENT 1 PARENT 2 PARENT 3*

*Firstly, the child print statements are run before the parent print statements because the parent has a wait(NULL) which forces the parent process to wait until the child finishes.*

*Secondly, the child simply takes the nums array and multiplies each value by a negative version of it.*

*Thirdly, the parent simply prints each number in nums.*

*Finally, data structures are not shared between the child and parent process thus each has its own copy of nums.*