**Himanshu Ojha**

**CS4414 (F2016) Operating Systems**

**Homework 1**

**Spring, 2016**

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

Solve the following problems. Each problem is worth 1 point.

(1) Including the initial parent process, how many processes are created by the program shown below? Show the result for each iteration in the for loop. (You will receive no point without the intermediate steps)

#include <stdio.h>

#include <unistd.h>

int main() {

int i;

for (i = 0; i < 4; i++)

fork();

return 0;

}

(2) Explain the circumstances under which the line of code marked printf("LINE J") in the following program will be reached.

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

int main() {

/\* fork a child process \*/

pid = fork();

if (pid < 0) { /\* error occurred \*/

fprintf(stderr, "Fork Failed");

return 1;

}

else if (pid == 0) { /\* child process \*/

execlp("/bin/ls", "ls", NULL);

printf ("LINE J");

}

else { /\* parent process \*/

/\* parent will wait for the child to complete \*/

wait(NULL);

printf("Child Complete");

}

return 0;

}

*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) Using the program shown below, identify the values of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively)

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

int main() {

/\* fork a child process \*/

pid = fork();

if (pid < 0) { /\* error occurred \*/

fprintf(stderr, "Fork Failed");

return 1;

}

else if (pid == 0) { /\* child process \*/

pid1 = getpid();

printf ("child: pid = %d", pid); /\* A \*/

printf("child: pid1 = %d", pid1); /\* B \*/

}

else { /\* parent process \*/

pid1 = getpid();

printf ("parent: pid = %d", pid); /\* C \*/

printf("parent: pid1 = %d", pid1); /\* D \*/

wait(NULL);

}

return 0;

}

*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) Using the program shown below, explain what the output will be at lines X and Y. (You will get no point if you just write down the result without explanation)

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

#define SIZE 5

int nums[SIZE] = {0, 1, 2, 3, 4};

int main() {

int i;

pid\_t pid;

pid = fork();

if (pid ==0) {

for (i = 0; i < SIZE; i++) {

nums[i] \*= -i;

printf("CHILD: %d ", nums[i]); /\* LINE X \*/

}

}

else if (pid > 0) {

wait(NULL);

for (i = 0; i < SIZE; i++)

printf("PARENT: %d ", nums[i]); /\* LINE Y \*/

}

return 0;

}

*The output will be:*

*CHILD 0 CHILD -1 CHILD -4 CHILD -6 CHILD -8 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.*

Use the following criteria to prepare and hand your homework in.

(1) Your homework can be generated using a word processor or by handwriting (but legible).

(2) Your homework should be submitted in class unless it is turned in late, which should be submitted via **collab** within three days from the due date in order to receive partial credit.

(3) Write down the following information on the first page of your homework: your name, homework number (i.e. homework 1), course number (i.e. CS4414), course title (i.e. Operating Systems), semester of the course (i.e. Spring, 2016), and your honor pledge.