Mat Ritchie

CS420

HW2

1) Including the initial parent process, how many processes are created by the program shown below? Briefly explain how you arrived at your answer.

#include <stdio.h>

#include <unistd.h>

int main()

{

/\* fork a child process \*/

fork();

/\* fork another child process \*/

fork();

/\* and fork another \*/

fork();

return 0;

}

Because this code doesn’t interact with its children, the parent waits until each child has finished and disappeared before launching the next. The parent creates 3 children in addition to itself for a total of 4 processes in existence, but only a maximum of 2 processes exist at any one point in time.

2) Using the program 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.) Explain how you arrived at your answer.

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

int main() //ID 2600

{

pid\_t pid, pid1;

/\* fork a child process \*/

pid = fork(); //ID 2603

if (pid < 0) { /\* error occurred \*/ //no execute

fprintf(stderr, "Fork Failed");

return 1;

}

else if (pid == 0) { /\* child process \*/ //no execute

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;

}

If the child’s ID is 2603, the else-if won’t execute, making A and B n/a, so the parent program would end up in the else block, making C a value of 2603 and D a value of (I’m guessing since it doesn’t say in the slides) 2604.

3) Using the program shown below, EXPLAIN what the output will be at Line A. That is, don't simply write down what the output is. Determine what the output is, and explain how you arrived at your answer.

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

int value = 5;

int main()

{

pid\_t pid;

pid = fork();

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

value += 15;

return 0;

}

else if (pid > 0) { /\* parent process \*/

wait(NULL);

printf("PARENT: value = %d\n", value); /\* LINE A \*/

return 0;

}

}

Because of the nature of the program, the first if-block must not execute, so when the checking step of the else-if block happens, the ‘value’ is still at 5. This block does not change ‘value’ so at Line A, ‘value’ = 5.