CSE 3430 **Lab 3** **AUTUMN 2021**

Name: Michael Napoli

Due: Sunday 12/5/21 11:30 pm on Carmen

The goal of this CSE 3430 lab is to give students an opportunity to explore the Unix/Linux fork() system call, which is used to create processes.

NOTE: You must compile and run the code for this lab on stdlinux – NO EXCEPTIONS!
ALSO NOTE: You should type your answers in this document, and then save it and export it as a pdf, and then submit it to Carmen.

BEFORE YOU DO THE STEPS BELOW, read the file Linux fork.pdf on Carmen in the Labs folder.

In your directory for CSE 3430, create a lab3 directory. From that directory, run this command:

$ firefox https://carmen.osu.edu/#

When firefox starts and opens Carmen, log in, and go to the Carmen page for our course.

From Files > Lab > Lab 3, download the forktest1.c file.

Now, do the following (Be sure to enter your name at the top of this document where you see **Name:** , above. Your answers to the questions below should be typed into this document below, wherever you see **ANSWER:**).

Next, compile the forktest1.c program using the ***gcc*** compiler, with the command below

$ gcc forktest1.c -o forktest1

1) Now run the forktest1 program: $ forktest1

Briefly describe what the program does (What output does the program produce)?

ANSWER:

The fork program splits the program into two separate subroutines. These subroutines operate independently over the same variables. They can be identified by their output in the command window. The program ends when both processes are complete.

2) How many times do you have to run the program to get a different ordering of the output lines (It should typically be no more than about 5 times)?

ANSWER:

I ran the program 4 times to get a different ordering.

3) How many processes were running for the forktest program (the maximum number running at any time – only processes related to forktest, and not independent processes created by other programs) when the program forktest ran? Explain how you got the answer.

ANSWER:

2 processes were running. Both function1 and funtion2 were completed once. Also, the fork() command is only called once, meaning only two processes are running.

4) At what point in the execution of forktest does a child process get created? When the child process is created, explain all of the things which the OS does.

ANSWER:

The child process is created when the fork() command is called. When this happens, the fork returns either -1, or the pid of the parent, and if the parent is being executed, the fork returns 0. Then the process continues in two parallel running programs. (for this example)

5) Consider the variable *num* which is declared in the program, and the values that are printed out for the variable in one run of the program (you only need to consider one run for this question).
   a) Are all the values which are printed out for the variable the same?

ANSWER:

No. All the values for num are not the same.

   b) If the values printed out for *num* are not all the same, explain, as clearly as you can, ***why*** they are different.

ANSWER:

When the child process is created, it creates a copy of the main program in its own individual block of memory. Here the value of num is also created. When function1 and function2 do calculations on the value of num, they are actually working on two different variables who share the same name with respect to their blocks of memory.

6) Give the pid of the parent process for ONE RUN of the program.

ANSWER:

pid for parent: 4295

7) How did you know the pid of the parent (Do not simply say that the program prints it out; how do you know the pid belongs to the parent?)?

ANSWER:

Since function1 runs when the return_value variable is equal to zero, it means that function1 corresponds to the child of the parent function. Which means that the parent process runs when the return_value is anything larger than zero, and function2 is executed. The above pid comes from the function2 output.

8) Give the pid of the child process for THE SAME RUN of the program.

ANSWER:

pid for child: 4296

9) How did you know the pid of the child (Do not simply say that the program prints it out; how do you know the pid belongs to the child?)?

ANSWER:

The child function receives a return_value of 0 from the parent function. Since function1 is executed when the return_value is equal to zero, the output from function2 corresponds to the child process. The above pid comes from the output of function2.

10) What can you say about the relationship of the pid of the parent, and the pid of the child?

ANSWER:

The pid of the child is used so that it knows the location of the memory block that child and can switch between them when necessary. The child gets a return_value (pid) of zero, because the parent is the original running process, and returning to the parent is to return to the most recent iteration of memory for the main program, in which case the location in memory is already saved.

TO SUBMIT YOUR WORK: Fill in your answers above, save the document, and export it as a pdf. Then, submit it to Carmen for Lab 3.

Labs not submitted by the due date and time, but within 24 hours after, will be assessed a 25% late penalty of the grade for the whole assignment; labs are not accepted after that time.