CSE 2431 HOMEWORK 1 SU 2020
KEY-GRADER

68 points total

Online Submission to Carmen Due: Thursday 5/21/20 ***at 11:30 pm***.

Question 1. How many processes are created when the program runs (include in your count the process created to run the program when you execute it from the command line on stdlinux). Describe briefly what is done by the process(es); refer to parent and child process when you describe what the processes do..
ANSWER: Two processes are created; the parent process forks a child, and then prints integer values from 1 to 8 , and the child process forked by the parent prints integer values from 9 to 16.

a) Are the results what you expected? Explain.
ANSWER: [Any reasonable answer accepted, as long as concepts related to processes and scheduling mentioned.]

b) Run the program various times, until the sequence of lines which print values (what is output as "value") is different. How many times did you have to run the program to get a different sequence?
ANSWER: [Any reasonable answer accepted, as long as concepts related to processes and scheduling mentioned.]

c) Now explain, as clearly as you can, in terms which mention the CPU scheduler and I/O system calls, why the sequence can vary for different runs of the program.
ANSWER: [Answer should mention OS CPU scheduler or I/O scheduler, and that the CPU has a single core.]

Question 2. How many processes are created when the program runs (include in your count the process created to run the program when you execute it from the command line on stdlinux)?
ANSWER: Two processes are created, a parent, which forks a child, and the child created by fork.

a) What value or values is/are output by the program for the variable *num*? Answer in terms of parent and child processes.
ANSWER: The num value output by the parent process is 10; the value output by the child is 15.

b) The value of *num* is changed from the initial value by the program code. Is this change shown whenever the value of num is printed in the output? Answer in terms of parent and child processes.
ANSWER: No, the change from 10 to 15 is not shown in the output of the parent; it is only shown in the output of the child.

c) Explain why the change in *num* is or is not reflected by the output (depending on what you answered for Question b) above). Give your explanation in terms of parent and child processes.

ANSWER: The change is not reflected in the output of the parent because the change is made by code executed by the child; therefore, the change is only shown in the output of the child, but not in that of the parent.

Question 3. How many processes are created when the program runs (include in your count the process created to run the program when you execute it from the command line on stdlinux)?
ANSWER: Two processes are created, a parent, which forks a child, and the child created by fork.

a) What value or values is/are output by the program for the variable *num*? Answer in terms of parent and child processes.
ANSWER: The num value output by the parent is 8; the num value output by the child process is 18.

b) The value of *num* is changed from the initial value by the program code. Is this change shown whenever the value of num is printed in the output? Answer in terms of parent and child processes.
ANSWER: No, the change from 8 to 18 is not shown in the parent's output; the change from 8 to 18 is shown in the output of the child.

c) Explain why the change in *num* is or is not reflected by the output (depending on what you answered for Question b) above). Give your explanation in terms of parent and child processes.
ANSWER: The change is not reflected in the output of the parent because the change is made by code executed by the child; the change is reflected in the output of the child, because the change is made in the child's code.

Question 4. How many processes are created when the program runs (include in your count the process created to run the program when you execute it from the command line on stdlinux)? [**DO NOT just count pids!** You need to justify this answer with your explanation for part a) below.]
ANSWER: A total of 5 processes are created when the program runs.

a) Explain the parent-child relationships between the processes created when the program executes. You can make a diagram if it is helpful (but it is not required), but you need to give a verbal (word-based) description of the relationships. Explain, as clearly as possible (and clarity counts!) how the execution of the C code in forktest4.c results in the creation of processes which have the parent-child relationships you identify. If the terms are needed to in order to answer, for a child of a child, you can use the term grandchild, and for a child of a grandchild, you can use the term great-grandchild.
ANSWER: We can use numbers to describe the 5 processes and their relationships. When the program is executed, a process is created which runs the program as a process, P_1, which is a parent that forks a child, P_2, in main.

P_2 calls a function Code1 which forks a child, P_3. When P_2 returns from the function Code1, it calls function Code1 a second time, which forks another child, P_4. When P_3 returns from function Code1, it also calls function Code1, which forks another child, P_5. The parent-child relationships are shown below:

Process	**Child or Children**
P_1	**P_2**
P_2	**$P_{3,}$ P_4 (grandchildren of P_1)**
P_3	**P_5 (grandchild of P_2)**

Question 5. What do you think the point was of entering and running these programs as processes? What did you learn?

ANSWER: The student's answer should include two points:

-When processes run on the CPU or do I/O is controlled by the OS; [3 points]

-Each process has its own address space, and it does not have access to code or data in the address spaces of other processes.

Question 6. What is a **pid**? How is a **pid** useful? It appears that a child process has two pids associated with it (Note: This does not mean a child process *has* two pids, but rather, it has two pids associated with it); what are the values of the two pids? Why are there two?

ANSWER: A pid is a unique (non-negative) number which identifies each process in the system. The pid is useful because it is used by the system to identify the process. A child process has two pids associated with it, because one, the pid of the child, identifies the child itself, but the other, the ppid, or parent process id, identifies the parent process of the child, that is, the one which executed fork to create the child.

Question 7. Look at the online manual page for **ps**, that is, do a **'man ps'**. What does **ps** do? Enter and run the commands '**ps**', '**ps –f**'', and '**ps –af'**. What are the differences?

ANSWER:

ps: is a command that gives information about a subset of the current active processes in the system, including the pid of each process. By default (that is, used with no options), it gives a report of various information about active processes which are associated with the terminal from which the ps command is run, and which are associated with the userid of the user who runs the ps command.

ps -f: This command outputs more detailed information about the same processes which are displayed by ps; in addition to pid, this option also displays the ppid and the uid (or userid) which is associated with the process, and time information for the process.

ps -af: This command shows the same processes as ps -f, except it does not display session leaders (parent processes of running processes) or processes not associated with a terminal.

Question 8. Look at the online manual page for **kill**. What does **kill** do? How might it be useful? How does it relate to the **ps** command?

ANSWER: kill is a command that can be used to send a signal to a process. By default, it sends a TERM (termination) signal; this can be used with the pid of a process to terminate the process.

kill relates to ps because ps can be used to get the pid of a process, and then kill can be used to kill the process.

Directions for this part: Give a one or two sentence description, *in your own words*, of each of the following system calls:

Question 9. fork

ANSWER: fork() is a system call which can be used to create a child process, which will be identical to the parent process, except that the child will have its own pid, different from the parent's pid. fork returns the pid of the child to the parent, but returns 0 to the child.

Question 10. getpid

ANSWER: getpid() is a system call which returns the process id (pid) of the process which calls it.

Question 11. getppid

ANSWER: getppid() is a system call which returns the process id (pid) of the parent of the process which calls it.

Question 12. wait

ANSWER: wait() is a system call which can be used by a process to wait (in effect, suspend its execution) until one or more other child processes terminate(s).

Type your answers to the questions above in a document with a word processor, and export the document to a pdf file called hw1.pdf; to submit the file hw1.pdf to Carmen, log in, and select HW1 under Assignments.

Homework not submitted by the deadline on the due date but within 24 hours will be assessed a 25% late penalty of the grade for the whole homework assignment; homework is not accepted after that time.