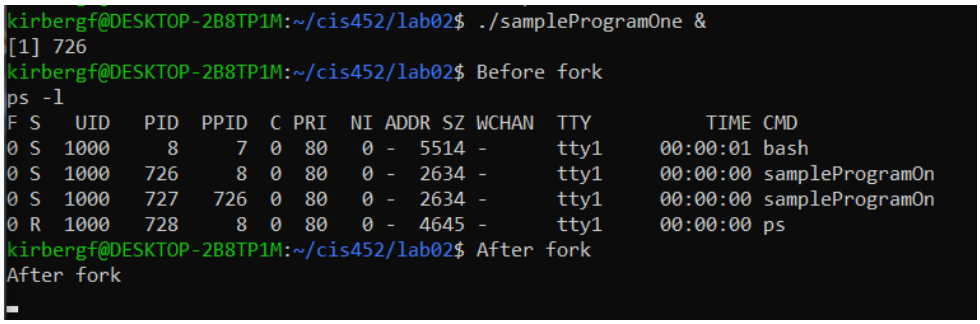
Fabian Kirberg
Sebastian Torrejon
Lab2
Sep 9, 2023

1. Compile and run Sample Program 1, how many lines are printed by the program?(2pts)
3 lines are printed by the program

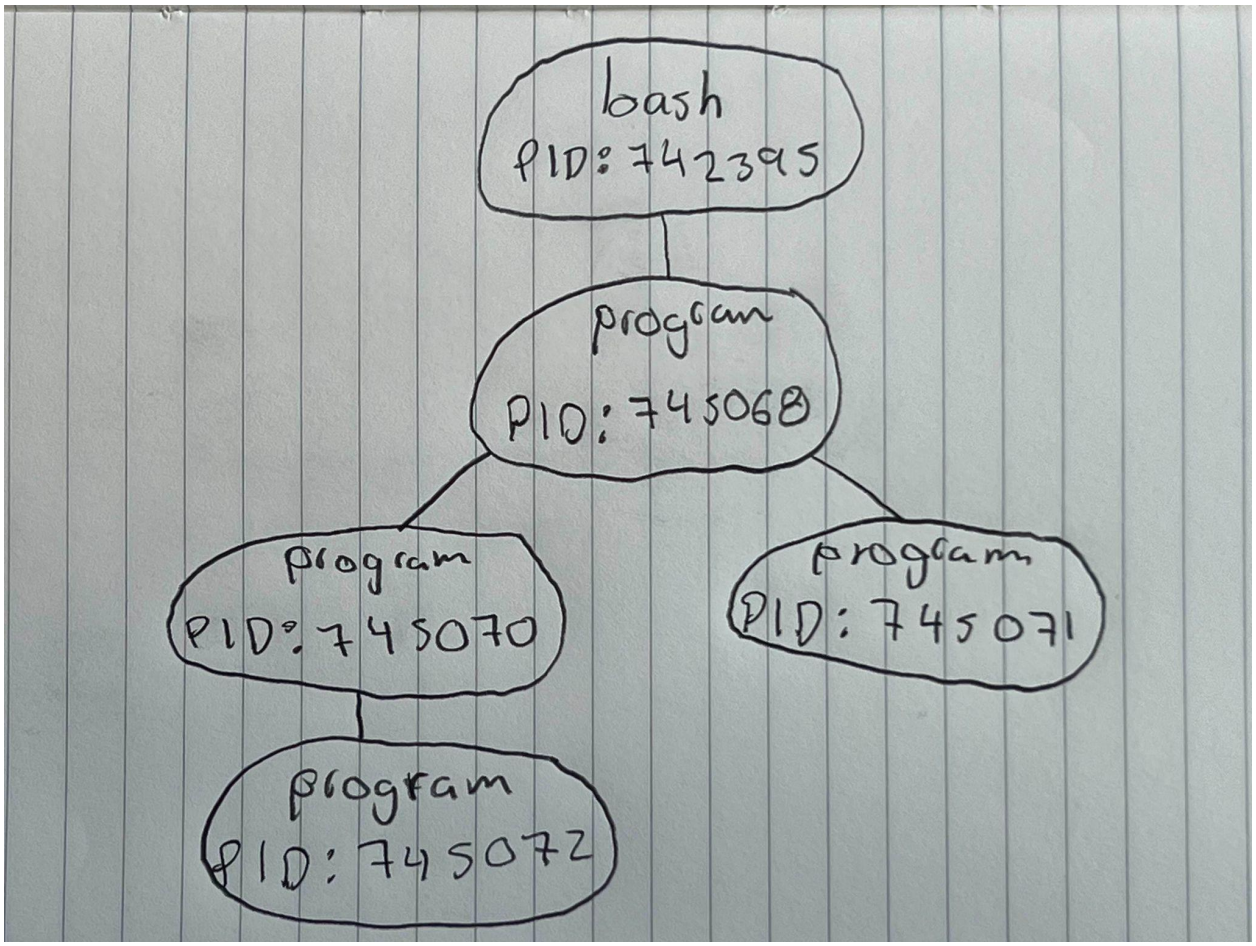
2. Describe what is happening to produce the output observed (4 pts)
The program runs normally until it reaches fork(), after which the process is cloned and another, separate instance of it runs parallel to it. This results in printf("After fork") being run twice in total.

3. Insert a 10-second call to the function sleep() after the fork in Sample Program 1 and recompile. Run Program 1 in the background (use &). Consult the man pages for the ps (process status) utility; they will help you determine how to display and interpret the various types of information that is reported. Look especially for "verbose mode" or "long format". Then, using the appropriate options, observe and report the PIDs and the status (i.e., execution state info) of your executing program. Provide a brief explanation of your observations. (4 pts)



S: Indicates the current status of the process.
PID: The unique identifier for each process.
PPID: The ID of the parent process.
CMD: The name of the command or program being executed.
The first process **(PID: 8)** is the shell being used to run the programs.
From that process **(PID: 8)**, the sampleProgramOne is as a second process **(PID: 726)**.
The second process **(PID: 726)** has a PPID: 8 because it is the child of the first process **(PID: 8)**. Then the fork() is called and it duplicates the second process **(PID: 726)**
The third process is created **(PID: 727)** from the fork() with a PPID: 726 since it is the child of that process.
The 4th and final process **(PID: 728)** is the ps command being used to determine the above and it has a PPID: 8 since that command is applied to the whole program which is the process with PID: 8.
Additionally, the shell process and the two sampleProgramOne processes have a status code of S, which means that they are sleeping. This is because of the sleep(10) line in the program. The ps process has a status code of R, because it is currently being run.

4. Create a diagram illustrating how sampleProgramTwo executes (i.e., give a process hierarchy diagram (see figure 3.8)). Run the program several times with small input values (e.g., 2...5) to help you understand exactly what is happening (5pts).



5. In the context of process state, process operations, and especially process scheduling, describe what you observed and try to explain what is happening to produce the observed results (5 pts).

The program begins by checking if there are enough arguments given. If there are not enough, the program exits.

(process state) Then, the program assignments the first additional argument to the variable limit as an integer and then forks into 2 processes. Next these two processes each fork again, resulting in a total of 4 processes. Forming a tree-like structure.

(process operation) Each of these 4 processes run concurrently and print their PID # and run a loop that prints 0 to limit-1.

(process scheduling) The scheduling of processes is determined by the operating system's scheduler, which decides which process to run when on the CPU. The order in which these processes execute and the interleaving of their execution depend on the scheduler's decisions. Since they run concurrently, the output lines from different processes may interleave unpredictably.

6. Provide the exact line of code that you inserted for the wait() system call (2pts)

```
child = wait(&status);
```

7. Which prints first, the child or the parent? Why? Describe the interaction between the exit() function and the wait() system call. You may want to experiment by changing the value to better understand the interaction (6 pts).

The child prints first.
The child will go into the ‘else if’ block of code, and the parent will go into the ‘else’ block of code. The parent code will encounter a wait() call that makes it wait for the child to terminate before it goes to the next line of code and prints something. The child doesn’t need to wait for anything and prints something. So the child will print first and then terminate. Once it terminates, it will return the status of 0 and the parent that was waiting for that, will continue and then finally print something.
The exit() is used by the child process to terminate itself, and the wait() in the parent process is used to retrieve the termination status of the child process.

8. When is the second print line ("After the exec") printed? Explain your answer. (3pts)

The After the exec will never be printed. If the execvp is successful, it will ignore any following code and just run whatever executable we called, like the command we gave the program (‘ls’, ‘date’) so it will never reach that print statement. Also if that execvp fails, it will print the error and then exit, so it never reaches the ‘After the exec’ print statement

9. Explain how the second argument passed to execvp() is used (3pts)?

The second argument is a pointer to an array of pointers to null-terminated strings, which means you can give multiple command-line arguments and it will point to an array of all of these arguments. It basically allows you to call the command with multiple arguments in it.