

## Program 2 – Spawning a Process

### Due: Friday, February 23 @ 11:59pm

#### Part I

You are to modify your sleepy program (program 1) as follows:

- Name this program *twoSleepy*.
- When *twoSleepy* is called using *twoSleepy n*, it will create a child process that is an exact duplicate of itself using the *fork* function. If *fork* returns a 0 you will need to jump to the child code. If *fork* returns a non-zero value you will jump to the parent code. You will need to do a little research to fully understand how this works. You will also need to research and use the *exit* and *wait* functions to ensure that the child process ends before the parent process.
- Both parent and child will use the same value of *n* to determine the number of seconds each process should run. Just like in program 1, implement this timing by looping *n* times of *sleep(1)* - this will put the process to sleep for one second *n* times before exiting.
- In each loop print out the process ID, the parent's process ID, and the loop count so that that particular process can be identified. The process ID can be obtained from the *getpid* function. The parent process ID can be obtained from the *getppid* function.
- Each line of output produced by the original process should be in the format:

"Original Process with PID: xxxxxx and PPID: yyyyyy tick *n*"

Where xxxxxx is the process ID, yyyyyy is its parent process ID, and *n* is the tick count.

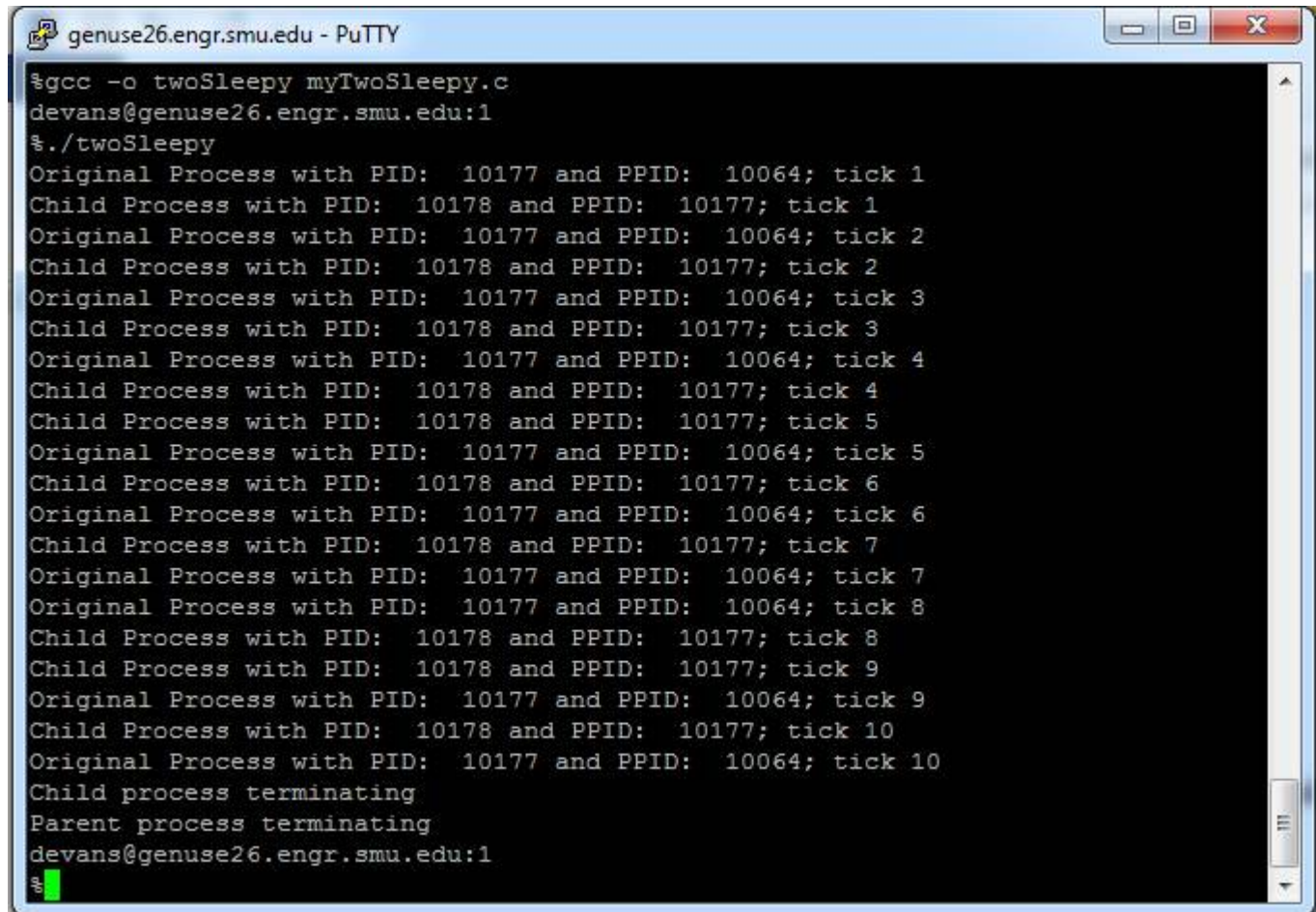
- Each line of output produced by the child process should be in the format:

"Child Process with PID: xxxxxx and PPID: yyyyyy tick *n*"

- Each process should also display a termination message.
- Run the program several times with different values for *n*.
- Your program will need the following header files:

```
#include <unistd.h>
#include <sys/types.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/wait.h>
```

Sample Run:



```
genuse26.engr.smu.edu - PuTTY
$gcc -o twoSleepy myTwoSleepy.c
devans@genuse26.engr.smu.edu:1
$./twoSleepy
Original Process with PID: 10177 and PPID: 10064; tick 1
Child Process with PID: 10178 and PPID: 10177; tick 1
Original Process with PID: 10177 and PPID: 10064; tick 2
Child Process with PID: 10178 and PPID: 10177; tick 2
Original Process with PID: 10177 and PPID: 10064; tick 3
Child Process with PID: 10178 and PPID: 10177; tick 3
Original Process with PID: 10177 and PPID: 10064; tick 4
Child Process with PID: 10178 and PPID: 10177; tick 4
Child Process with PID: 10178 and PPID: 10177; tick 5
Original Process with PID: 10177 and PPID: 10064; tick 5
Child Process with PID: 10178 and PPID: 10177; tick 6
Original Process with PID: 10177 and PPID: 10064; tick 6
Child Process with PID: 10178 and PPID: 10177; tick 7
Original Process with PID: 10177 and PPID: 10064; tick 7
Original Process with PID: 10177 and PPID: 10064; tick 8
Child Process with PID: 10178 and PPID: 10177; tick 8
Child Process with PID: 10178 and PPID: 10177; tick 9
Original Process with PID: 10177 and PPID: 10064; tick 9
Child Process with PID: 10178 and PPID: 10177; tick 10
Original Process with PID: 10177 and PPID: 10064; tick 10
Child process terminating
Parent process terminating
devans@genuse26.engr.smu.edu:1
$
```

- Using a simple five-state process model, what states do you think each process goes through during its lifetime? Might this explain why the output is so evenly interleaved between parent and child? Document your answers using in-line comments.

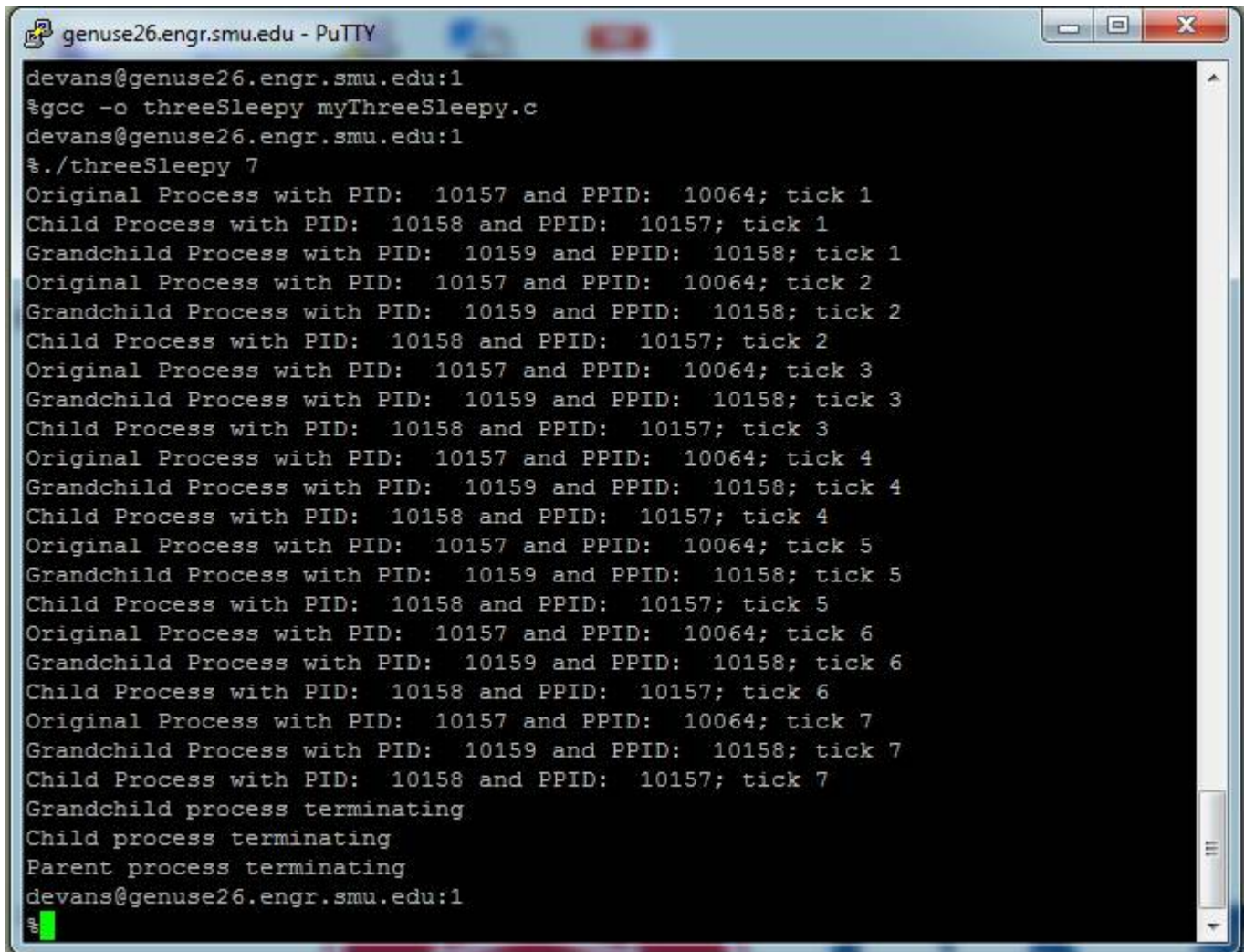
## Part II

- Comment out the call to the *sleep* and *wait* functions in the parent. This will ensure that the parent terminates before the child. All other code will remain the same. Recompile and run your program several times.
- Describe what you see using comments in your source code. Is there anything unusual about the PPID in the child? Do some research and explain what is happening.

## Part III (optional)

- Modify your program from Part I so that the child process calls *fork* to create a child of its own. Use *exit* and *wait* to ensure that the processes end in reverse order from which they were created.

Sample Run:



```
genuse26.engr.smu.edu - PuTTY
devans@genuse26.engr.smu.edu:1
%gcc -o threeSleepy myThreeSleepy.c
devans@genuse26.engr.smu.edu:1
%./threeSleepy 7
Original Process with PID: 10157 and PPID: 10064; tick 1
Child Process with PID: 10158 and PPID: 10157; tick 1
Grandchild Process with PID: 10159 and PPID: 10158; tick 1
Original Process with PID: 10157 and PPID: 10064; tick 2
Grandchild Process with PID: 10159 and PPID: 10158; tick 2
Child Process with PID: 10158 and PPID: 10157; tick 2
Original Process with PID: 10157 and PPID: 10064; tick 3
Grandchild Process with PID: 10159 and PPID: 10158; tick 3
Child Process with PID: 10158 and PPID: 10157; tick 3
Original Process with PID: 10157 and PPID: 10064; tick 4
Grandchild Process with PID: 10159 and PPID: 10158; tick 4
Child Process with PID: 10158 and PPID: 10157; tick 4
Original Process with PID: 10157 and PPID: 10064; tick 5
Grandchild Process with PID: 10159 and PPID: 10158; tick 5
Child Process with PID: 10158 and PPID: 10157; tick 5
Original Process with PID: 10157 and PPID: 10064; tick 6
Grandchild Process with PID: 10159 and PPID: 10158; tick 6
Child Process with PID: 10158 and PPID: 10157; tick 6
Original Process with PID: 10157 and PPID: 10064; tick 7
Grandchild Process with PID: 10159 and PPID: 10158; tick 7
Child Process with PID: 10158 and PPID: 10157; tick 7
Grandchild process terminating
Child process terminating
Parent process terminating
devans@genuse26.engr.smu.edu:1
%
```