COMP 2560 Winter 2024 — Lab 7

Question 1

Check out the manual page for the function *strsignal(. . .)* to see what it does and how to use it. Write a small C program using the *strsignal(. . .) function* to find out how many signals are supported by the Linux system running on the CS server (you can loop 100 times to try signal numbers from 0 to 99). You could also use the command "kill -l" to find out the signals supported on the CS server to verify the output from your program.

No submission is required for this question.

Question 2

This lab question is related to Assignment 4 Question 2 and should be helpful. Do the following:

1. In your main function, using the signal () function, write code to ignore ctrl+z and install a signal handler for ctrl+c, the handler simply displays "ctrl-c pressed!" when invoked.
2. Test run your program so far in (a) to make sure it works as expected.
3. Now, fork a child process, and let the child process run the program "donothing.c" (code posted in the past). In the parent process's code, loop for 15 iterations, and in each iteration, sleep for 1 second, as below.

```
for(i=1; i<=15; i++){
    printf("I am in parent process.\n");
    //send a signal to child
    sleep(1);
}
```

   The comment above in the for loop is the placeholder to use the kill() function to send a signal to either the parent or the child process to test how they respond to the ctrl+c and ctrl+z signals.

Completing the above 3 steps will give you a solid start for solving Question 2 in Assignment 4.
No submission is required for this question.

Question 3

Although we did not spend time in the lecture to study signals SIGCONT and SIGSTOP, it is not hard at all to see how these two signals work. Read the brief description of these two signals in the textbook on pages 317 and 320. Then follow the steps below to create your program to observe how these two signals work.

1. In the main function, fork a child process.
2. In the child process, use an infinite loop to print out something. For example, you can print out the value of an int variable and each iteration increases the value of this int variable. You may also optionally sleep for a second or two in the loop to slow down the loop.
3. In the parent process, sleep for a few seconds first, then send the SIGSTOP signal to the child process.
4. In the parent process, again sleep for a few seconds, then send the SIGCONT signal to the child process.
5. Lastly, in the parent process, you can send the SIGINT signal to the child process to terminate it.

When running your program, observe what happens in steps 3 and 4. Is the child process stopped and continued upon the receipt of the signals?

Submission is required for this question at the end of your lab session. Please submit your source code and the script files (learned in Lab 2) showing you compile and run your program, and indicate which lab section.