COMP 2560 Fall 2020

Lab 7

IMPORTANT CHANGE REGARDING VIDEO SUBMISSION

When video submission required, please submit a shareable link to your video file (use google drive, OneDrive, YouTube, etc.), do not submit the video file itself.

The purpose of this lab is to practice signal processing.

As in Lab 6, there is one question that requires submission at the end of your lab session. The rest of the questions are due on Nov. 22, 11: 59 PM.

Please put in your name and lab section as comments on top in your source code.

---

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 strsignal(…) function to find out how many signals are supported by the Linux system running on the CS server.

   Submit your source code , the script files, and explain/show how many signals you find supported by the Linux system running on the CS servers as comments in your source code.

2. We briefly discussed the signal SIGCHLD, and the following claim is from the course slide.

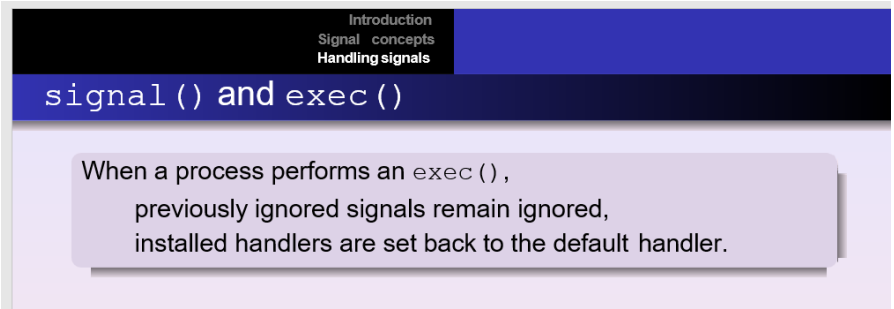
SIGCHLD	Whenever a process terminates or stops, the SIGCHLD signal is sent to the parent. By default, this signal is ignored, so the parent must catch this signal if it wants to be notified whenever a child's status changes. The normal action in the signal-catching function is to call one of the wait functions to fetch the child's process ID and termination status.

   Design your own program to demonstrate what is described above. Here is an idea.
   a) In the main function, first install a signal handler for the SIGCHILD signal.
   b) The signal handler for SIGCHLD should use wait or waitpid function to retrieve the status of the child process, and print out the status value retrieved (we discussed a few macros to retrieve exit status when studying the wait and waitpid function before the midterm).

c) Then the parent process fork a child process.
d) In the child process, simulating the child process terminates normally by doing the following: the child process sleeps for 5 second, then terminates it by calling the exit(…) function.
e) Observe if the handler for the SIGCHLD signal works as expected.

Submit your source code and the script files.

3. The following claim is from the course slide.



In order to verify this claim, write a program as instructed below.

(a) In your main function, using the signal () function, write code to ignore "ctrl+z", and install a signal handler for "ctrl+c", the handler may simply display "ctrl-c pressed!".

This step is to set up a scenario that your program will ignore the signal generated by "ctrl+z", and your program has a user-supplied signal handler for "ctrl+c".

We want to see later , as the slide claimed, after exec(…), if the ignored signals ("ctrl+z") remain ignored and the installed handler ("ctrl+c") are set back to the default handler.

(b) Test run your program so far in (a) to make sure it works as expected for "ctrl+z" and "ctrl+c".

(c) Now, fork a child process, and let the child process run the program "donothing.c" (demonstrated in class during lecture and code is posted). In the parent process's code, loop for, say 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 response to the “ctrl+c” and “ctrl+z” signals. We will study how to send a signal to a process on Monday Nov. 16.

Completing the above (a), (b), and (c).

Submit your source code and the script files.