COMP 2560 Winter 2024—Assignment 4

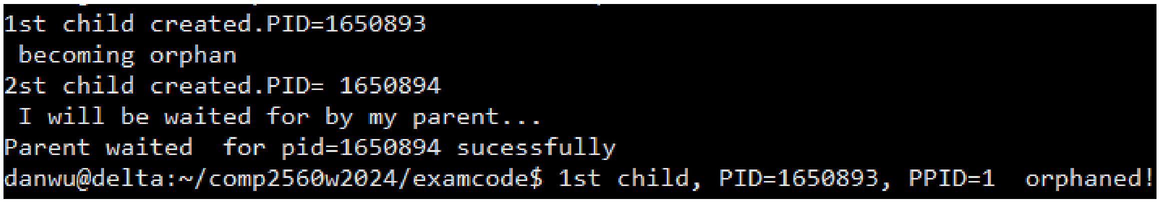
Due: 11: 59 PM, Mar. 17

Question 0

Please submit the bash scripting LinkedIn course completion certificate that you should have obtained during the reading week.

Question 1

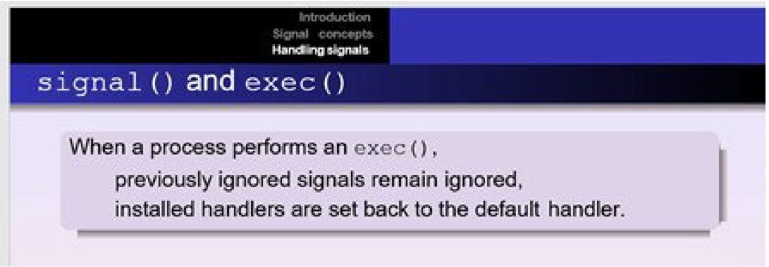
Write a C program where the parent process generates exactly two child processes such that a) the parent process uses the waitpid(..) function to wait for the second child process and prints its pid and its exit status value (of your choice), and b) the first child process becomes an orphan. The output of your program should be similar to the one shown below:



```
1st child created.PID=1650893
 becoming orphan
2st child created.PID= 1650894
 I will be waited for by my parent...
Parent waited  for pid=1650894 sucessfully
danwu@delta:~/comp2560w2024/examcode$ 1st child, PID=1650893, PPID=1  orphaned!
```

Question 2

This question is related to a slide we (will) discuss in class, which is shown below. Verify its claim by designing a program to demonstrate that


Introduction
Signal concepts
Handling signals

signal() and exec()

When a process performs an exec(),
previously ignored signals remain ignored,
installed handlers are set back to the default handler.

Figure 1: Signal and exec().

a) in the parent process, you install a handler for (ctrl+c) and you ignore ctrl+z.

b) the parent process then forks a child process.

c) inside the child process, using execXX() function to run a simple program such as "donothing.c" we discussed in class, and check to see if the handler you installed (for ctrl+c) in the parent is set back to the default handler, and check to see if ctrl+z is still ignored. (Hint: you could use the kill() function to send signals to process to test the claims.)

Explain briefly how your program is designed to test the claims **as comments** in your source code. Script the output from running your program with the timing option.

Question 3

Write a program simulating an alarm clock. In the main function, you fork a child process, and the child process sleeps for 5 seconds (the number of seconds is an command line argument, see below for a sample run), after which the child process sends the signal (SIGALRM) to its parent process. The parent process, after forking the child process, pause, upon receiving the SIGALRM signal, process prints out a message "Ding!". The following is a sample run

```
$./alarm   5
  alarm application running
  waiting for alarm to go off
  <5 second pause>
  Ding!
  Done!
```

Question 4

This exercise shows the idea of how to use one user-defined signal handler to handle two signals. The two signals chosen for this exercise are SIGINT and SIGQUIT. Read (or Google) the descriptions of these two signals on page 319 and page 320 in the textbook. Note this idea applies to other signals as well. What you need to do is to write your own one signal handler function and install this handler for both SIGINT and SIGQUIT. When your signal handler is invoked, print out the signal no. (which is the argument to your user-defined signal handler function) so that we know which signal has been caught.

Submission Requirements

For all questions, you need **to submit your source code**. Also, use the script command with the timing option to record you compiling and running your program and submit the script file and timing file for each question as well. **Record a short video with time stamps** for each question to explain your code design and compile and run your code. Please properly name all the submitted files.