CSCE 313 # QUIZ 3 Fall 2021

Total Points 100

Q1 [15 pts]: Consider the program below and answer the following questions with proper explanation.

A. What is the output? How much time does the program take to run?[10 points]
B. What is the output with line 5 commented? How much time will it take now? [10 points]

```
1 void signal_handler (int signo){
2   printf ("Got SIGUSR1\n");
3 }
4 int main (){
5   signal (SIGUSR1, signal_handler);//comment out fr b)
6   int pid = fork ();
7   if (pid == 0){// child process
8       for (int i=0; i<5; i++){
9          kill(getppid(), SIGUSR1);
10         sleep (1);
11      }
12  }else{ // parent process
13      wait(0);
14  }
15}
```

a) We create a signal handler function and attach it to SIGUSR1. Then the parent creates a child, and goes into a wait, while the child calls kill() 5 times. The kill function simply sends the SIGUSR1 signal to the parent process. So the parent simply prints “Got SIGUSR\n”. 5 times at 1 second intervals.

Output: 5 seconds to run
Got SIGUSR1
Got SIGUSR1
Got SIGUSR1
Got SIGUSR1
Got SIGUSR1

b) Default action of SIGUSR1 is to terminate the process, so the parent would create a child and the child would immediately send the SIGUSR1 signal to the parent which is to kill the process resulting in immediate execution with no output.

Q2 [20 pts]: Do you think the following program can possibly produce the output: **"2 children are done"** in the sense that the my_handler() would run only twice? If No, why not? If Yes, why? Please explain your answer.

```
int num_children_done = 0;
void my_handler (int sig){
     ++num_children_done;
}
int main (){
    // install handler
    signal (SIGCHLD, my_handler);
    // create 5 child procs,
    for (int i=1; i<=5; i++){
        int pid = fork ();
        if (pid == 0){
            sleep (5);//all die simultaneously
            return 0;
        }
    }
    sleep (10);
    cout<< "Total " <<  num_children_done<<" children are done"<<endl;
}
```

YES

First we install the handler that basically increases the num_children_done variable when the children process finishes.

Then we create 5 children and set them all to sleep for 5 seconds.

Then the parent sleeps for 10 seconds.

In these 10 seconds all 5 of the children finish execution simultaneously and call the SIGCHLD which increments the num_children_done variable.

Only 2 of these SIGINTS are processed because the signals all arrive simultaneously so signal blocking is initialized and once the first signal is processed it only moves on to the second signal but not the other 3.

In the end **"2 children are done"** is printed, even though all children are done because only 2 signals were processed.

Q3 [20 pts]: The following program counts the number of primes in a range of numbers accurately given enough time. Now, since this program takes a long time, the user may get impatient and press Ctrl+C to kill it. The default behavior for that is immediate termination of the program without any output. Your task is to change this program by giving it a “best-effort” ability, meaning that it will print out the count of however many prime numbers it could find before receiving the SIGINT signal generated by the Ctrl+C. The count will obviously be inaccurate; however, it is the best given the limited time. In summary, you will write a signal handler for SIGINT that will: a) print the partial count and b) terminate the program after that. You are not allowed to use multiple threads, or other processes.

```
int num_primes = 0; // holds # of primes
void count_primes (int start, int end){
      // check each number in range [start, end] for primality
      for (int num = start; num <= end; ++num) {
            int i;
            for (i = 2; (i <= num) && (num % i != 0); ++i)
                  ;
            if (i == num)  // if no divisor found, it is a prime
                  ++num_primes;
      }
}

void my_handler(int sig){
      cout << “Found “ << num_primes << “ prime numbers in the range”<<endl;
      exit(sig);
}

int main (int ac, char** av){
      signal (SIGINT, my_handler);
      count_primes (1, 1000000); // count all primes <= 1 million
      cout <<"Found "<<num_primes<<" prime numbers in the range"<<endl;
}
```

CSCE 313 # QUIZ 3 Fall 2021

Q4 [45 pts]: Implement a program that takes as argument a path, absolute or relative, to a file or a directory and produces the same output as the command “ls -l” does. If no argument is provided, your program should default to the current directory - just like the builtin “ls -l” command. You cannot use any exec() function for this program. Test your program on directories that have at least one of each of the following: regular files, directories, soft links, fifo files, write/read/execute protected files. Your program’s output must match exactly with the builtin command - including text and formatting (except colors and highlights). For instance, your program must follow a soft link and show which actual path it points to - just like the “ls -l”. The following man page can be your starting point:

g++ main.cpp -o ls

./ls -l <file path>

For some reason the dates for modified and created directories or files show as the current time and date and I did not have time to go back and see the issue.

For some reason when I readlink to get the path to the symbolic link, the buffer carries some extra garbage at the end and again did not have time to go back and fix it.

The spacing is also pretty weird because I implemented the symbolic link stuff till the very end so the spacing there is going to be off