

TDQC

Signaler

Jack Spence

11 Jan 2019

1. Write-Up

1.1 Requirements

Requirements were to create a program that will print prime numbers, aprox. 1 per second, and output them to the screen, while allowing the behavior of the print to be modified by Operating System signals being sent to the program.

1.2 Suggested Features

Suggested feature attempted in my submission is to implement the “-r” flag to start in reversed order, and the “-e” flag to stipulate where the primes should stop printing.

Usage:

1) make || make profile || make debug

2) ./bin/signaler [-r] [-e <end #>]

2. Project Design Plans

2.1 Initial Design Plans

My initial design plan was to use a doubly linked list to store the primes and generate only primes that were near the one being printed. Initially thinking the prior 100 and next 100 primes would be a good starting point. I knew the process of generating prime was going to be rather quick, but didn't want to waste a lot of memory by generating a super huge number then printing them. I thought doubly linked would be good as well, because if the reverse call came in printing in reverse would be trivial.

2.2 What didn't work

At first I was using the `sleep(3)` function to cause the 1 second delay. What I found with this method was that anytime a signal was sent the program would print immediately. I then looked into other means and found the `alarm(2)` system call and `pause(2)` to catch the `SIGALRM` signal which told my loop to print a prime.

2.3 What went well

The implementation of the doubly linked list of primes was an easy solution. There could have been other ways, such as using an array of characters and using bits as positions, that would have had less overhead, but even after generating primes constantly for 20 seconds or so, I only used about 5k of RAM.

2.4 Conclusion

This project at the beginning looked more like a math problem with being able to solve the prime number debacle, but turned out to be a decent introduction to signal handling in C. Prior to this I had only handled Exceptions in python so being able to have similar functionality in C is pretty good.

3. Test Procedures

3.1 Testing

Testing of signaler was pretty simple because the functionality was rather simple.

3.2 Testing Procedure

MAKE- No Errors

```
skydev@skydev:~/TDQC-Projects/signaler$ make
cc -Wall -Wextra -Wpedantic -Waggregate-return -Wwrite-strings -Wvla -Wfloat-equal -std=c11 -c -o src/signaler.o src/signaler.c
cc -Wall -Wextra -Wpedantic -Waggregate-return -Wwrite-strings -Wvla -Wfloat-equal -std=c11 -c -o src/primes.o src/primes.c
cc -Wall -Wextra -Wpedantic -Waggregate-return -Wwrite-strings -Wvla -Wfloat-equal -std=c11 src/signaler.o src/primes.o -o bin/signaler
skydev@skydev:~/TDQC-Projects/signaler$
```

-r Test

```
skydev@skydev:~/TDQC-Projects/signaler$ ./bin/signaler -r
2
skydev@skydev:~/TDQC-Projects/signaler$
```

-e <NUM> Test

```
skydev@skydev:~/TDQC-Projects/signaler$ ./bin/signaler -e 10
2
3
5
7
skydev@skydev:~/TDQC-Projects/signaler$
```

-e <GARBAGE> Test: In this case end is set to 0, which is effectively INF

```
skydev@skydev:~/TDQC-Projects/signaler$ ./bin/signaler -e asdfasdf
2
3
5
7
11
13
17
19
^C
skydev@skydev:~/TDQC-Projects/signaler$
```

Receiving SIGHUP – Restart Count of primes

```
skydev 0 • 1 bash
skydev@skydev:~/TDQC-Projects/signaler$ kill -s SIGHUP $(ps -elf | grep './bin/signaler' | grep -v color | awk -F" " '{ print $4 }')
skydev@skydev:~/TDQC-Projects/signaler$

29
31
37
41
43
47
53
59
61
67
2
3
5
^C
skydev@skydev:~/TDQC-Projects/signaler$
```

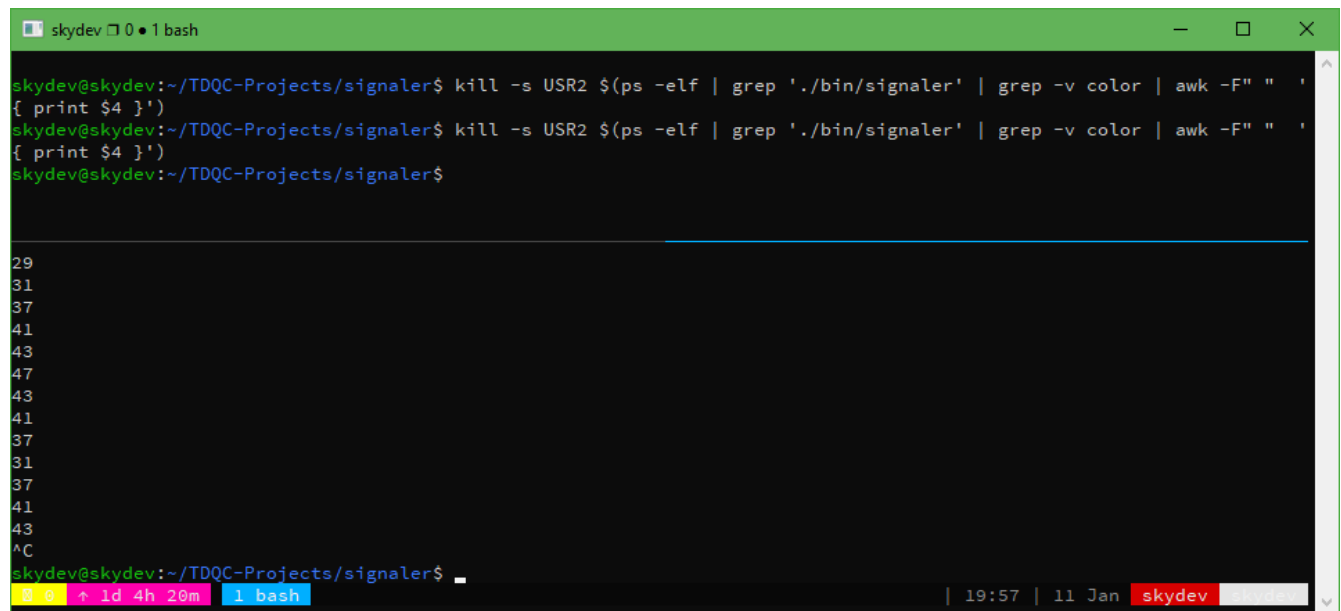
Receiving SIGUSR1 – Skip Next prime(67 in Example)

```
skydev 0 • 1 bash
skydev@skydev:~/TDQC-Projects/signaler$ kill -s USR1 $(ps -elf | grep './bin/signaler' | grep -v color | awk -F" " '{ print $4 }')
skydev@skydev:~/TDQC-Projects/signaler$ ^C
skydev@skydev:~/TDQC-Projects/signaler$

29
31
37
41
43
47
53
59
61
71
73
79
83
^C
skydev@skydev:~/TDQC-Projects/signaler$
```

← MISSING

Receiving SIGUSR2 – Change count direction



```
skydev@skydev:~/TDQC-Projects/signaler$ kill -s USR2 $(ps -elf | grep './bin/signaler' | grep -v color | awk -F" " '{ print $4 }')
```

```
skydev@skydev:~/TDQC-Projects/signaler$ kill -s USR2 $(ps -elf | grep './bin/signaler' | grep -v color | awk -F" " '{ print $4 }')
```

```
skydev@skydev:~/TDQC-Projects/signaler$
```

```
29
```

```
31
```

```
37
```

```
41
```

```
43
```

```
47
```

```
43
```

```
41
```

```
37
```

```
31
```

```
37
```

```
41
```

```
43
```

```
^C
```

```
skydev@skydev:~/TDQC-Projects/signaler$
```

0 9 ↑ 1d 4h 20m 1 bash | 19:57 | 11 Jan skydev