Name: Sean Bartholomew
Date: 20160506
Current Module: Operating Systems
Project Name: Signaler

Project Goals:

The project is designed to calculate prime numbers sequentially. It is also intended to catch signals and handle them as well as handle optional arguments.

Considerations:

How to process signals
What algorithm to use to calculate prime numbers
How much memory will the program have access to.
How to pass the state data to the signal handler.

Initial Design:

The program is implemented entirely in one file: signaler.c. It is composed of three functions. There is one global variable: a structure which contains the last prime number, a maximum number, and the direction to go to calculate the next one. The global variable is necessary to pass data to the signal handler. There are three functions: main, handler, and next_prime. The main function supports optargs, and initializes the signal handler. The signal handler processes signals appropriately. The next_prime function calculates the next prime number in the direction indicated by the global variable.

Data Flow:

After initializing the global structure and setting up the signal handler, main enters a forever loop, calling next_prime. And printing the result to screen. Upon receiving a signal from the user, handler will make appropriate changes to the data. The next_prime loop will continue generating prime numbers until uint_MAX is reached.

Communication Protocol:

CMD line options:
-s start at the next prime number greater than n
-r emit decreasing values from number n, incompatible with -s
-e exit the program if a prime number greater than n would be printed
Signals handled:
SIGHUP Restart emitting prime numbers from 2.
SIGUSR1 Skip the next prime number to be emitted.
SIGUSR2 Reverse direction; go from emitting increasing to decreasing, or vice versa.

Potential Pitfalls:

The largest pitfall will be ensuring that the prime numbers are generated correctly.

Test Plan:

User Test:

Run the program with no options.
Run program with invalid options.
Run program with valid options and invalid parameters.
Run the program with each option individually.
Run the program with multiple options at the same time.
Send all signals to the running program.
Run valgrind.

Test Cases:

Test starting at a number less than max.
Test starting at a number larger than max.
Test starting at two and decreasing.
Test starting at a large int and decreasing.
Test signal input.

Expected Result:

I expect the behavior outlined in the initial design to be appropriately implemented. Additionally I expect that the program will have no memory leaks. The program will not seg-fault.

Conclusion:

The best implementation for the prime number generator ended up being the brute force method. As there was no way of knowing how long the function would be running, it was impractical to attempt to implement a sieving function. The signal handler was not terribly difficult. It did however necessitate the use of a global variable. In the future I would rather put the global variable in a separate file and utilize getter and setter functions to access it.