Operating Systems Writeup

Name: Joshua Abraham

Date: 2 November 2017

Current Module: Operating Systems

Project Name: "Relay"

Project Goals:

The relay project aims to create two programs that communicate via Interprocess Communication. This IPC will be one-way, from a "**dispatcher**" program that takes input from standard input to a "**listener**" program that prints text sent to the first program.

Considerations:

- **Dispatcher** must take text from stdin and exit when it receives an EOF.
- **Listener** should exit if the listener is closed or not running.
- Multiple **listeners** should be able to run simultaneously from different users/directories.

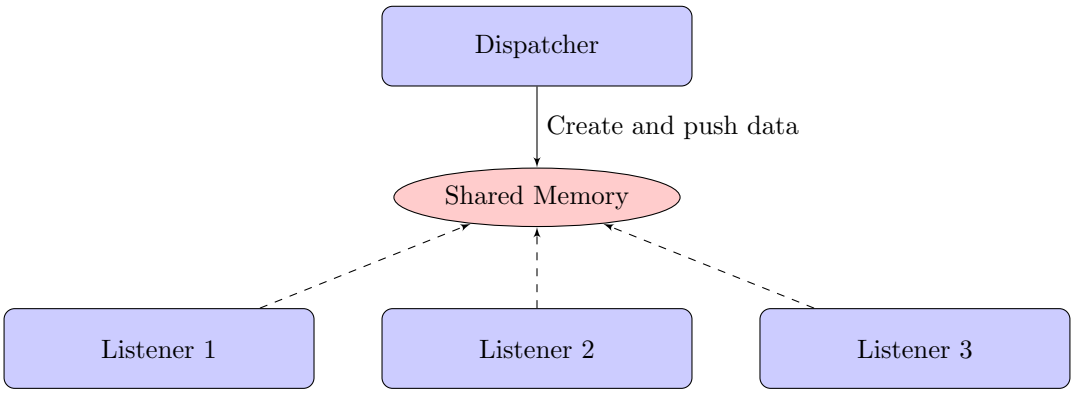
Initial Design:

The project is composed of the following files:

· *Makefile*: The main makefile for the project.

Data Flow:

The data flow from **dispatcher** to **listener** is as follows:



(1)

The basic flow of the signaler program is shown in figure 1. The signaler gets its command line arguments and sets the appropriate flags and values. It enters its main loop and sets its signal handlers. Signaler spawns a timer thread to pause its execution for .8 seconds. It then prints the timestamp. Lastly, signaler generates the next prime number in the sequence to be printed. We do so with Fermat's little theorem, seen below: We perform 4 rounds of the Fermat Primality Test. This is a very fast and performant calculation. While this might be overkill for relatively small integers, this method will pay off when calculating extremely large primes.

Communications Protocol:

The signaler and sender communicated via signals. We specifically use the SIGHUP, SIGUSR1, SIGUSR2, and SIGKILL.

Potential Pitfalls:

- Prime generation taking too much time.
- Incorrectly handling signals.
- Pausing execution without sleep.

Test Plan:

User Tests:

· Ran signaler and sender simultaneously.

· Ran signaler and sent signals with bash kill command.

Test Cases:

· Unit tests for the prime number generation and Fermat's Little Theorem.

Conclusion:

The signaler project has been challenging and a learning experience. I learned about signals and threads while working on this project. Pausing execution without sleep was a fun challenge to solve. While writing the signaler, I ran into bugs that were difficult to debug and required me to use GDB extensively. One example was resetting the signal handler every time a signal was received. If I had more time to work on the project I would improve it by implementing a more robust prime number generator such as Miller-Rabin. Overall, it was a success and I am satisfied with what I wrote.