Jay Offerdahl
EECS 678
Lab 4 Report

1. There are two special signals (KILL and STOP) which are not handled by the process they are sent to. When a KILL or STOP signal is generated, the operating system itself handles this signal and kills or stops the appropriate process. Considering what you learned in today's lab, speculate as to why the system designers chose to include signals which are handled solely by the operating system.

- I believe that the operating system is the only handler of KILL and STOP signals because this would prevent programs from running forever without being able to be killed. If processes were able to handle KILL and STOP commands, they could potentially keep themselves alive even though the operating system might want them to die.
- This would be especially bad because malicious software that needs to stay alive despite the user and operating system's request would be able to. Also, if programs became unresponsive and needed to be terminated, if they had handlers for this signals, they would be unable to be killed.

2. What benefit do we gain from using the pause system call as opposed to an infinite while loop?

- The pause system call waits for input, so instead of entering an infinite loop of nothing (which still works, ish), we can loop through each time we wait for input to the program. This saves resources in the computational machine as there's not constant processing when the program is waiting.

3. Why do we mask other signals while inside the signal handler?

- The goal of our program was to interact with the Ctrl-C SIGINT, Ctrl-Z SIGTSTP, and SIGALRM signals. We only wanted these signals to be sent while inside the signal handler, so we blocked the rest of them using the signal mask. The purpose of this is to block other signals from sending until our current signal handler has completed. This is also known as a race condition, which is exactly what we're trying to avoid.

4. When we implement the timeout, we do not mask the SIGALRM signal. Why?

- We don't mask the SIGALRM signal because for the functionality of our program, we have to send an alarm signal when inside the signal handler for Ctrl-C. If we masked SIGALRM, the timeout would wait until the Ctrl-C signal was fully processed. However, this would mean the program would always exit because no cancel-alarm signal was ever sent.