Gehrig Keane
2727430
EECS 678: Lab 4 SIGNALS

1. The necessity for end-all-be-all signals handled only by the operating system aligns swimmingly with common mistakes made on part of developers who try to handle and manipulate general operating system signals. An example with regard to this lab: early in experimentation after I'd implemented the signal handler, but before implementing the logic for termination testing the program resulted in a program whose termination was quite difficult. The process needed to be killed directly. An abstraction of this process extends to any low level system developer whose mistakes yield indeterminate programs. Ergo, it would be silly for an operating system to NOT have a mechanism of control above the program it's supposed to be controlling.

2. The pause() system call allows the operating system to move the program out of focus until a signal is received thereby allowing more productive computation time. The operating system can schedule more productive tasks in the meantime. A continuous while loop without the pause() directive remains in focus and the CPU remains busy with meaningless computation.

3. Signal masking allows the program to redefine signal behavior and manipulate their behavior within the program. Where a normal ^C signal would halt a programs execution the signal mask allow the program catch said signal.

4. We want the SIGALRM signal to break the pause() and terminate the program if the user doesn't respond to the continuation prompt. Masking the SIGALRM signal would essentially render the usefulness of said signal nil.