**CPE 435 Operating System Labs**

**Lab 5: POSIX Signals**

There are many mechanisms through which the processes communicate and in this lab we will discuss one such mechanism: Signals. Signals inform processes of the occurrence of asynchronous events. In this lab we will discuss how user defined handlers for particular signals can replace the default signals handlers and also how the processes can ignore the signals. By learning about signals, you can "protect" your programs from Control-C, arrange for an alarm clock signal to terminate your program if it takes too long to perform a task, and learn how UNIX uses signals during everyday operations.

Write a program in which a parent process creates a Child process:

* If the user sends ctr+c to the parent process before 10 seconds from start running, then the parent catches the signal and a message indicates that the system is protected.
* After 10 seconds, the parent process could be terminating using *ctr+c*, before the parent terminates it must send a signals to the Child. The Child then prints its process id and terminates.

Some useful system calls:

*int pause ( )*

pause ( ) suspends the calling process and returns when the calling process receives a signal. It is most often used to wait efficiently for an alarm signal. pause ( ) doesn't return anything useful.

*int kill(pid, sig)*

The command kill sends the specified signal to the specified process or process group. If no signal is specified, the TERM signal is sent. The TERM signal will kill processes which do not catch this signal.

pid is the process-ID of the process to receive the signal;

sig is the signal number. The effective user-IDs of the sending and receiving processes must be the same, or else the effective user-ID of the sending process must be the super user.

If pid is equal to zero, the signal is sent to every process in the same process group as the sender. This feature is frequently used with the kill command (kill 0) to kill all background processes without referring to their process-IDs. Processes in other process groups (such as a DBMS you happened to have started) won't receive the signal.

If pid is equal to -1, the signal is sent to all processes whose real user-ID is equal to the effective user-ID of the sender. This is a handy way to kill all processes you own, regardless of process group