Signaling

1. Explore signal.h and different types of signal macros.

- The **signal.h** header defines a variable type sig_atomic_t, two function calls, and several macros to handle different signals reported during a program's execution.
- **sig_atomic_t :** This is of int type and is used as a variable in a signal handler. This is an integral type of an object that can be accessed as an atomic entity, even in the presence of asynchronous signals
- **SIGABRT:** (Signal Abort) Abnormal termination, such as is initiated by the abort function.
- **SIGFPE:** (Signal Floating-Point Exception) Erroneous arithmetic operation, such as zero divide or an operation resulting in overflow (not necessarily with a floating-point operation).
- **SIGILL:** (Signal Illegal Instruction) Invalid function image, such as an illegal instruction. This is generally due to a corruption in the code or to an attempt to execute data.
- **SIGINT:** (Signal Interrupt) Interactive attention signal. Generally generated by the application user.
- **SIGSEGV:** (Signal Segmentation Violation) Invalid access to storage: When a program tries to read or write outside the memory it has allocated.
- **SIGTERM:** (Signal Terminate) Termination request sent to program.

2. Explore signal and kill library functions.

- The **kill()** function shall send a signal to a process or a group of processes specified by **pid**. The signal to be sent is specified by sig and is either one from the list given in <signal.h> or 0. If sig is 0 (the null signal), error checking is performed but no signal is actually sent. The null signal can be used to check the validity of pid.