To run the program do **make.**
To remove executable files do **make clean.**

In main. c, parent process forks 3 child processes namely s1, sr, and st.
S1 handles signals for SIGTERM which are given by kill(SIGTERM, pid_s1) command.
To share strings between s1 and e1 a pipe is used, to pass strings from e1 to s1's signal handler which prints these strings.
E2 has two signal handlers one for SIGALRM and the other for SIGTERM .

E1 and e2 are called in sr and st processes respectively by the command
execl("e1", "e1", str, NULL);
execl("e2", "e2", str, NULL);
In which the pid of s1 is passed which is used later to send SIGTERM signals to s1.

E1 and e2 both use setitimer() to produce SIGALRM signals which are send to e1 and e2 both, the duration in which singals are sent can be changed by changing #DEFINE INTERVAL 1000.

Both e1 and e2 have signal handlers that handle these signals differently.

E1 used RDRAND instruction with inline assembly to produce a random number generated by the CPU which is sent to s1 using a pipe and to also send a SIGTERM signal to s1.

E2 uses RDTSC instruction to produce the of cpu timestamps which are then converted to the current date and time using systems cpu frequency(here it's taken to be 1800MHz)
Also, this signalHandler for e2 produces SIGTERM to e2 which is handled by another signal handler which prints a string containing current date and time .
The program runs indefinitely till its stopped by sending SIGINT signal or any other signal.

Error Handling:
Various errors like fork() failing or signalhandler are not able to catch the signals produced are handled by errno and perror.

To print inside signalHandler printf is not used instead write system call is used to print.