Assignment 2_2

Files -

- S1.c
- helperS1.c
- ST.c
- helperST.c
- SR.c
- helperSR.c
- Makefile

Instructions to run -

```
make
```

S1.c & helperS1.c

- Creates 2 child processes using fork, inside the parent one again fork is called for SR Process and inside SR's Parent again fork is called for ST.
- Inside the SR parent SR.c's object file is executed and inside the ST parent ST.c's object file is executed using execl()
- Inside the first fork both signal handlers and created, sigaction and sa handlers are used to set siginfo to transfer info and the output generated is sent to stdout using write syscall
- The S1 child keeps running till it's interrupted by the user be entering Ctrl+C via the terminal
- The processes keep running due to waitall() function at end
- All these functions are defined inside the helperS1.c file

SR.c & helperSR.c

- The main SR.c file has been abstracted and internal implementation has been moved to helperSR.c to showcase only the major 4 components involved in SR.c which are getting pid from args, calling SRSignalCheck explained later, useitimerval explained later and waiter which is essentially a while(1) call which creates a loop
- useitimerval() is used to set timer while SRSignalCheck() is used to check for any errors and if none calls SRSignals() which generated a random number and sets the sival_int attribute
- InLineRandomNumberGenerator() function generates the random number using inline assembly's rdrand function which generates random numbers using Intel Chip Tech, it was added to AMD chips as well in 2015
- itimerRealerror() in helperSR.c checks for errors while using itimerval()

ST.c and helperST.c

- The main ST.c file has been abstracted and internal implementation has been moved to helperST.c to showcase only the major 5 components involved in ST.c which are getting pid from args, calling

sigalrmcall explained later, useitimerval explained later and waiter which is essentially a while(1) call which creates a loop

- sigalrmcall checks for errors and if none calls TimeHandler() which inturn calls printTime() which calls InLineTimeStampGenerator() which returns number of clock cycles since the CPU started. Dividing it by clock frequency gives the number of seconds. Since my System is an AMD Ryzen 5 4600HS it's CPU Clock Speed is 2994.385 MHz which is what the returned value is divided by to get time which is then sent via sigqueue to S1 where inside helperS1.c we have printTime() function which extract the hours minutes and second out of it and prints it via write syscall to stdout
- useitimerval() is used to set timer while itimererror() is used to check for any errors and if none allows the time to be set as implemented

Sigqueue Overall Functionality

- Sigqueue is used to send signals between different processes in our cases it is numbers which are being sent since sending char* is not possible