OS LAB4

In this assignment we implement alarmx() function with ROP technique we call function which was passes as parameter.

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

 248880672 bytes of free memory.  Free list:
           [0x002801E0 to 0x0EFD8FFF]
           [0x0FDEF000 to 0x0FDEFFFF]
    108589 bytes of Xinu code.
           [0x00100000 to 0x0011A82C]
   1336776 bytes of data.
           [0x0011DDA0 to 0x00264367]


I'm K MEHER HASANTH and my username is kmeherha.
Test process running code of main():
-1:zcalled by alarmx
```

Case I
I have called 3 alarmx() calls.
The first registers the alarm in the range of NPROC and 2*NPROC.
The second registers the alarm in range of 2*NPROC and 3*NPROC
When the second alarmx() is called the (void (*prcbftn)(void)) is updated with the latest function it is executed using ROP technique.
For third alarmx() call we see that output as -1:z Which means that the program was allowed to have at most 2 alarms and returned SYSERR on executing the 3rd alarm.
The function my bigloop simply prints **" called by alarmx"**

Case 2:

```
I'm K MEHER HASANTH and my username is kmeherha.
Test process running code of main():
called by alarmx
```

We called sleepms() after alarmx() which stores the value of function pointer **(void (*prcbftn)(void))** in process table. executedetoure2() is executed in sleepms if function pointer is present which calls mybigloop. This prints **“called by alarmx”**

IPC

Single sender

```
I'm K MEHER HASANTH and my username is kmeherha.
Test process running code of main():
starting process sndB
send B returned with out - 1
starting process sndA
recv msg - message 1
recv msg from sender pid - 5
```

We send message “message -1” to receiver .

We create process called sndB() which calls sendx(). Sendx() writes txt message -1 in the receiver process buffer to receiver process.

SndA() is calls recievex() which reads the data passed from sendx() and prints it on the screen i.e **“message-1”**

CASE 2:

We create multiple senders and single receiver.

```
Test process running code of main():
starting process sndB
starting process sndC
starting process sndD
send B returned with out - 1
send D returned with out - -1
starting process sndA
send C returned with out - 1
recv msg - message 1
recv msg from sender pid - 5
recv msg - message 2
recv msg from sender pid - 6
```

Here we send message-1 using sndB since the receiver was initially empty it copied the message and printed the value in its buffer.

sndC notices that receiver is full and writes to its buffer. After consuming the message receiver consumes the value of the buffer and allows to write prints message-2.

The 3rd sndD processor notices that recievers buffer is non empty and prblocksendr is not 0 hence it returns -1