Description:

We had to create 5 processes. P1 , Enc1 , Channel , Enc2 , P2.

The objective of the assignment was to use semaphores and shared memory so that we can exchange messages between P1 and P2. (First from P1 to P2 then from P2 to P1 and so on and so forth until we give the message TERM).

How to run:

After using make to compile the processes ,open two terminals.
In the first one run the p1_enc1_chan with a single int argument from 1 to 99.
Eg: ./p1_enc1_chan 15
(This int is the probability that each individual character will be changed.)
Then in the other terminal run the p2_enc2
Now you can enter the first message to p1_enc1_chan and watch it appear on the other terminal after successfully going through each process.

How it works:

The message given by the user has to be passed from P? To Enc? where it will be encoded using MD5.

Then the checksum will be added to the message struct so that we can later verify whether or not the message was corrupted.

After going through enc? The process reaches the communication channel (channel process) where it might be corrupted . (The probabilty that each individual character will be corrupted is given by the user as an argument).Then the channel passes it on to the other enc process.

Now the other enc process uses the checksum and MD5 to verify whether or not the message was corrupted. If it finds that the message was indeed corrupted it asks for retransmission from the other enc,else if it determines that the message was not corrupted it gives back a confirmation signal to channel and the other enc and passes the message to its P? Process where it shown to the user.

After this it is the P? Turn to send the message and the other P process has to wait for it.

How to end the processes:

The processes end when any of the two P1/P2 successfully send the message TERM to the other process.

Deeper look:

I create 4 shared memories P1→Enc1 , Enc1→Channel , Channel→Enc2 , Enc2→P2 and a couple of semaphores for each shared memory. The message travels through these shared memories to reach its destination. The semaphore signals are used to start and stop each process at the right time.

Additionally after encoding the message I add a checksum to the message struct. This way the receiving enc can compare the MD5 hash of the current message with the checksum. If they are the same, that means that the message has travelled successfully, else it probably needs retransmission.

Lastly, each enc? when sending a message waits to receive that it was transmitted correctly and the proceeds to wait for the next message.

Extra files:

There are may extra files that help with the implementation of the project. Namely:
rand_change.c/h: Helps with the randomization of the strings/messages
sema.c/h : Helps with the creation and usage of semaphores
shared_create.c/h : Helps with the creation and usage of shared memory
messg_struct.c/h : The struct that we use to exchange the messages.
Makefile : To compile and link the project.

Misc:

I used C to create the project. The ide used was vscode. I have checked the project with valgrind and gdb. There is a makefile included for your convenience.