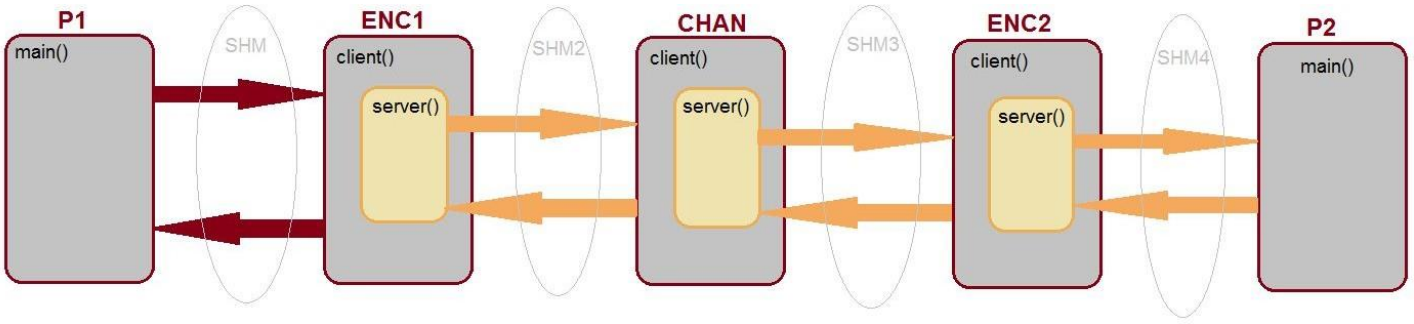
Varvitsiotis Athanasios

Operation System Communication Architecture

The Implementation of the 5 processes has been made in 5 different programs. So, we will have one process per program. In order to run my program, I run my Makefile (command: make) in a ssh terminal, then I open 4 more different ssh terminals (ie we will have total of 5 terminals) and I run each process separately, i.e. one in each terminal (commands: ./p1 ./enc1 ./channel ./enc2 ./p2 respectively).*In this order*. The process of inter-process communication in my five programs has been achieved by creating four shared memory segments which are framed by the appropriate semaphores. Memory segments and semaphore Keys are declared statically in each program with a define command. As far as semaphores are concerned, I have two functions, the mem_P, which performs the down operation and the mem_V , which performs the up operation. The mem_v and mem_p are called in such a way that there is no simultaneous write-read by the two communicating processes. Here is a painting of my implementation that illustrates the flow and the communication of the processes as well as the use of the client and server functions:



In process P1 we have the main() which exchanges messages with enc1. In process ENC1 we have the client() which exchanges messages with P1 and uses the server function (as an interface) which is responsible for communicating with the CHAN process. The CHAN and ENC2 processes operate with the same pattern. The P2 exchanges messages with the ENC2 process.

In addition, some more functions appear in the processes. We have the function free_resources which deletes the search memory segments and the semaphore segments between the two communicating processes. The addchecksumtomessage function is the one that extends the message string by adding the checksum to the real message. The checksumisModified function checks if the message has been altered by channel's noise. This function is used as a decryptor on enc1 - enc2 in order to check if the message sent has undergone a change. The noisefunction function distorts each character of the message (without taking into account the checksum) based on the probability that the user has given to the channel process (CHAN).