

CS 39006: Assignment 8
Asynchronous, signal driven I/O
Date: -11-April-2019
Due: In Class

In this assignment, you will write a simple UDP echo server using asynchronous, non-blocking I/O. The client takes a string as input from the keyboard and then sends it to the server (null-terminated). The server will receive a null-terminated message from the client, print it on display, and send the same message back to the client which then prints it on the display again. Assume max. size of a message is 100 chars including '\0'.

Normally, when you make an I/O call like accept, connect, recv/recvfrom, send/sendto etc. on a socket, if the call cannot be completed immediately (for ex., if recv is called but no data has arrived in the socket buffer), the call will block. So by default, such an I/O operation is a blocking I/O operation. However, it is possible to make such calls nonblocking, such that instead of blocking, the call will return immediately with an error and errno will be set appropriately. The program can then test for this, maybe do other things, and come back to do the I/O again later. In order to do this, the socket has to be made nonblocking. You have already done this in earlier assignments.

Another way to do this is signal driven I/O. In case of signal-driven I/O, the socket is made enabled for receiving a SIGIO signal, which is sent when certain conditions happen (incl. receive of data). The handler can then do the I/O, the main program can continue to do its own work.

To make a socket enabled for asynchronous I/O in this manner, you will need to first set some options in this order.

1. Set the owner of the socket to this process (can get the process id using getpid()). To do this, look up the F_SETOWN command in fcntl()
2. You have to make the socket ready for async I/O. To do this look up the O_ASYNC flag for the F_SETFL command in fcntl().

Of course you have to set up the signal handlers as usual.

Between using non-blocking flag as you have done before, and signal driven I/O, which one would you prefer and why? Write a comment at the TOP of your .c file submitted.

Submit two .c files, server.c and client.c in a single zip file <rollnumber>_Assignment6.zip or <rollnumber>_Assignment6.tar.gz.