File transfer using multi-channel stop-and-wait protocol

- **Client side**
    - Sender creates a child process and the parent and child create a TCP connection => open a channel
    - Both share file descriptor for the input file and hence if one reads some chunk of the file, the offsets automatically change in the other
    - They send the chunk read and now enter an infinite loop to wait for an ACK
    - They monitor the readability of the socket using select call
        - If select returns
            - 0 : the timeout has occured => ACK is lost and hence send the packet again
            - >0 : ACK has arrived
                - If ACK is corresponding to the packet sent, construct next packet to forward
                - Else out of order ACK has arrived, ignore this ACK and go in select loop again
- **Server side**
    - The server is a polling one.
    - Check readability on listening socket and adds the incoming connections to an array.
    - Also, check readability of the connections and when they are readable,
        - Receive the data packet
            - If the incoming sequence number is not expected one
                - Buffer the packet if space is available in queue
                - Else, reject the packet
            - Else,
                - Write this data to the file,
                - If there are any outstanding packets contiguous to this, transfer them to the file
        - Send an ACK back if packet is not rejected to same channel
- Instructions to run
    - Run `sh script.sh` in one terminal, this will create the executables and run the server
    - Run `./client` in another, this will run the client
    - Make sure you have an `input.txt` file in the current directory
    - After completion, following files will be created
        - destination_file.txt - File where server writes the output
        - client.log - Logs generated by client
        - server.log - logs generated by server
        - To view logs in sorted order run on a terminal following command
            - `sort *.log >> combinedLogs.log`
            - This will generate combined logs in sorted order of time in `combinedLogs.log`
- Change following parameters if required while testing
    - In server.h
        - Packet drop rate (PDR)
        - Number of out of order packets buffered (BUFFERSIZE)
        - server port number (SERVER_PORT)
        - server log file (SERVER_LOG_FILE)
        - destination_file (DESTINATION_FILE)
    - In client.h
        - input file (INPUT_FILE)
        - payload size (CHUNK_SIZE)
        - server IP (SERVER_IP)
        - server port number (SERVER_PORT)
        - timeout value (TIMEOUT_S)
        - max number of tries in case of not receiving ack (MAX_TRIES)
        - client log file (CLIENT_LOG_FILE)