TWITTER SIMULATOR

Work done in this project:

* Wrote a shell script to reap(kill) any zombie process in the system.
* Wrote a Tweet (Tweet.c) program which sends tweets(messages) to the Server program (Server.c) concurrently.
* Wrote a Server (Server.c) program which accepts tweets from Tweet (Tweet.c) program.
* The exchange of tweets between the Tweet and Server program is achieved using socket programming via TCP connection.
* Once Server receives the tweets, it forwards to Follower (Follower.c) program via UDP connection.
* As soon as Follower (Follower.c) program receives the tweets, it sends feedback to Server (Server.c) program concurrently via TCP connection.
* Finally the Server (Server.c) program forwards the feedback to respective tweet process in tweet (Tweet.c) program via TCP connection.
* Onscreen messages are displayed on the terminals to help understand the interaction between Server, Tweet and Follower program.

Code File:

* **zombie.sh**
  + Selects and reaps zombie processes if any using “preap” command
  + “awk”, “grep” and “tr” commands are used to filter the required set of zombie processes
  + Since zombie processes cannot be reaped before **60** **seconds using the preap command,** I use “sleep” command to ensure that the zombie processes are reaped after the 60 second period successfully
* **Tweet.c**
  + 3 concurrent tweet processes are created using “fork()” command and each process reads their respective “Tweet(A/B/C).txt” files.
  + Once each tweet process knows the message it needs to transmit, it creates and populates the structure “addrinfo”.
  + While populating, the server PORT number is statically assigned “3669” as per instructions, the protocol is set to TCP (SOCK\_STREAM) and the protocol family is set to “AF\_INET”.
  + The IP Address of each of the tweet process is obtained using “getaddrinfo()” function and passing “localhost” as one of its parameters.
  + TCP Socket is then created using “socket()” function and the socket descriptor is obtained.
  + TCP connection is then established using the “connect()” function.
  + Once connection is established the respective messages are sent by the tweet processes using “send()” function.
  + Once the message is sent the connection is closed using the “close()” function.
  + The dynamic TCP port for each of the Tweet process is obtained using the “getsocketname()” function.
  + Onscreen messages are displayed as per instructions and delay is used to ensure easy readability.
  + Once Server (Server.c) program receives the feedbacks from all the Followers, the Tweet (Tweet.c) program initiates the TCP connection with Server concurrently for 3 tweets using fork().
  + Each Tweet process(A,B,C) will receive their respective feedback from server.
  + Once all the tweet processes (A,B,C) receive their feedbacks, the tweet (Tweet.c) program exits.
* **Server.c**
  + Structure “addrinfo” is created and populated in the Server program.
  + While populating, the protocol is set to TCP (SOCK\_STREAM) and the protocol family is set to “AF\_INET”.
  + The server port is statically assigned “3669” and its IP is obtained using “getaddrinfo()” function and passing “localhost” as one of its parameters.
  + TCP Socket is then created using “socket()” function and the socket descriptor is obtained.
  + The created socket is bound to an IP and PORT using the “bind()” function.
  + The server then listens for any incoming TCP connection using “listen()” command. A queue size of 10 is used in this program.
  + Once the server accepts the connection using “accept()” command, I used “fork()” to create a child process, this is done to ensure that server can handle multiple TCP requests at a time.
  + The tweet(message) is then received from the Tweet process using the “recv()” function.
  + Once tweets from all 3 Tweet processes are received, the TCP connection is closed using “close()” function.
  + Onscreen messages are displayed as per instructions.
  + All tweets are stored locally using “mmap()” function, for Phase 2 purpose!
  + In Phase2, once all the tweets are received, they are forwarded to Follower Process via UDP connection using “sendto()” function.
  + Also note that an appropriate socket structure has been created (struct addrinfo DGRAM) to handle UDP connections.
  + Once Server receives the feedback from Follower (Follower.c) program using TCP connection, it stores them locally using “mmap()” function.
  + As soon as the Tweet (Tweet.c) program initiates the final TCP conection, the Server program forwards the respective feedbacks to respective tweets(A,B,C) in tweet program.
  + WaitFor() function is used to insert appropriate delays in code!
* **Follower.c**
  + Follower processes are given a static UDP port as per instructions.
  + First, each follower(1,2,3,4,5) process receives its respective tweets from the Server (Server.c) program via UDP connection using the above static UDP ports.
  + Structure “DGRAM” is used for this purpose.
  + The tweets are received using the “”recvfrom()” function.
  + As soon as the tweets are received each of the follower processes, will call their respective “tcp()” function.
  + It uses the static TCP port of Server as given in instructions.
  + Each Follower process opens their respective feedback files “Follower(1/2/3/4/5).txt“ and then sends 4 packets to Server via TCP connection.
  + The commands used to accomplish the same is similar to the TCP connections implemented earlier.
  + Once it sends all the feedbacks to the Server, the Follower (Follower.c) program exits.
  + “WaitFor()” function is used to insert appropriate delays in code! For easy understanding.
* **Makefile**
  + Compiles both the “Server.c” and the “Tweet.c” program
  + Also runs the script “zombie.sh” to reap/destroy any zombie process.
* **TweetA.txt TweetB.txt TweetC.txt**
  + Contains the tweets that needs to be transmitted from “Tweet.c” program to “Server.c” program.
* **Follower1.txt Follower2.txt Follower3.txt Follower4.txt Follower5.txt**
  + Contains the feedback that needs to be transmitted from “Follower.c” program to “Server.c” program.

Steps to run the Program:

1. In Terminal type “make” and wait for 60 seconds to ensure no zombie processes are running!. Although the zombie.sh might throw some errors/warnings, please ignore the messages and proceed to step 2!
2. Open 3 terminals.
   * In **Terminal 1** type “./Server”
   * In **Terminal 2** type “./Tweet”
   * In **Terminal 3** type “./Follower”
3. Wait for the Processes to complete. Might take some time!!
4. Press “Ctrl+C” in **Terminal 1** to stop Server Process.