CS 39006: Assignment 1

Simple Communication using Datagram Socket using POSIX C

Assignment Date: 03-Jan-2019

Deadline: 10-Jan-2019 2:00 PM

Objective:

The objective of this assignment is to get familiar with datagram sockets using POSIX C programming. The target is to establish a communication between two computers (processes) using datagram socket. A datagram socket uses a simple communication paradigm to transfer short messages between two computers (processes) without ensuring any reliability.

Problem Statement:

Your task will be to write two programs - one program corresponding to the server process and another program corresponding to the client process. The client process requests for the content of a file (by providing the file name) and the server process sends the contents of that file to the client.

For simplicity, we assume that the file is a simple text file that contains a set of words, with the first work being HELLO and the last word being END. We assume that HELLO and END are two special keywords that do not come anywhere except at the first line (HELLO) and the last line (END). The content of a sample file looks as follows.

```
HELLO
CAT DOG
TIGER
LION
HORSE ZEBRA COW
END
```

The transfer of the contents of the file works using a communication protocol as follows.

1. The client first sends the file name to the server.
2. The server looks for the file in the local directory, if the file is not there it sends back with a message NOTFOUND. By receiving this message, the client prints an error message “File Not Found” and exits.
3. If the file is present, the server reads the first line of the file, which contains HELLO, and sends this message to the client.
4. After receiving HELLO, the client creates a local file (a different file name from the requested one) and sends a message WORD1 to the server. This message indicates that the client is requesting for the first word.
5. After receiving the message WORD1, the server sends the first word (after HELLO) to the client. The client writes this word to the local file after receiving it and sends the message WORD2 to request for the next word. This procedure continues for each word.

6. On receiving WORDi, the server sends the i-th word to the client. This process continues until the client receives the keyword END.
7. Once the client receives the keyword END, it closes the local file after writing the last word to the file.

Submission Instruction:

You should write two C programs - wordserver.c (contains the server program) and wordclient.c (contains the client program). Keep these two files in a single compressed folder (zip or tar.gz) having the name <roll number>_Assignment1.zip or <roll number>_Assignment1.tar.gz. Upload this compressed folder at Moodle course page by the deadline (10th Jan 2019 2:00 PM).

Marks Distributions: (Total: 30)

Server Socket Program: 16

a) Define socket FD and call the socket function - 1
b) Create server address structure and initiate them - 1
c) Call the bind function - 1
d) Wait for the file name from the client (with recvfrom call) - 1
e) Initiate the client address length properly before the recvfrom call - 1
f) Receive the file name and parse from the buffer correctly as a string - 1
g) Open the file on receiving the file name (if the file exists) - 1
h) Send NOTFOUND message if the file is not there and exit the server program - 1
i) Read HELLO from the file and send it to the server - 1
j) Keep on waiting for the WORDi (WORD1, WORD2, …) message from the client - 1
k) Read the next word on receiving a WORDi message - 1
l) Send the word to the client - 1
m) If the word is END, send the word to the client and exit - 1
n) Close the file - 1
o) Close the server socket every time before exiting the program - 2 (there is likely to be at least two instances - one if the file is not found and other at the end)

Client Socket Program: 12

a) Define socket FD and call the socket function - 1
b) Create the address structures and initiate them - 1
c) Send filename to the server - 1
d) Wait for the HELLO message through recvfrom call - 1
e) On receiving HELLO, create the local file in write mode - 1
f) Send WORD1 message to the server - 1
g) Wait for an word from the server until END is received - 1
h) Receive the word - 1
i) Store the word in the file - 1

j) Send WORDi after receiving every word, keep on incrementing the counter i - 1
k) On receiving END, close the file - 1
l) On receiving END, close the socket and exit - 1

Additional: 2

a) The file at the server and the client generated file are exactly similar (all NULL characters has been transferred correctly) - 1
b) Readability of the code (indentation, commenting) - 1