Assignment 2

Multi-user chat system using Named Pipes

By Anmol Kumar (2018382)

For this assignment, I have implemented IPC between a server and two or more clients using named pipes. In general, ordinary pipes cannot be used for communication between unrelated processes. This can be achieved by using another kind of pipes called **Named pipes**.

Code Description

- There are two C files : server.c and client.c. For initiating the communication between different terminals where we first need to run server.c before executing two or more client.c because at least 2 clients are required for a message to be sent and received.

- The server would have the named pipe : addition_fifo_server. Each client would have its unique named pipe with the nomenclature : add_fifo_client(pid of the current terminal process). Both types of pipes are created in /tmp because it has appropriate permissions.
  This is done to avoid pipe ambiguity for any number of pipes. The piped is opened using mkfifo() command with appropriate permissions.

- Two new process threads, sending thread and receiving threads are created so that both sending and receiving functions are active during any time of execution of client.c.

- As soon as a process is created, a string is passed to the server named pipe which is parsed and hence the server identifies the creation of a new client. The server maintains a string array of named pipes of all clients. After that the client process can send or receive a message simultaneously.

- In send(), a string is passed which contains the sender's named pipe string, sending requirement(either send the message to a particular client or send a message to all client) and the message itself, the message is then parsed when received by the server.

- Then a new string is formulated which contains the receiver's named pipe name, sender's number and the message. The message is then read using read() function of any client which then prints the message.

- These processes are valid for any number of client processes, provided that a single server is online.

Compilation instructions

- The server.c file is compiled using the command `gcc -pthread -o server server.c`
- The client.c file is compiled using the command `gcc -pthread -o client client.c`
- We then first need to run the command `./server` and after that `./client` in any different number of terminals.

Expected Input

- Any client could either send the message to a single client or to all the clients. For single receiver client :

  Send (receiver number) (message to be sent)

  For all the receiver clients :

  Send all (message to be sent)

Expected Output

- When server.c is executed :

```
Server Online
```

- When client.c is executed :

```
Welcome to the Chat !
Creation Status: success
Your address is: /tmp/add_client_fifo2852
Your port number is: 2
```

- Whenever a client sends a message, it is displayed on server terminal as :

```
>RECEIVED message from 2: Hello I am Anmol !
```

- Whenever a server sends a message to the receiver client, it is displayed on server terminal as :

```
>SENT message to all: Hello I am Anmol !
```

- The received message is then displayed on the receivers' terminal as :

```
Message from 2: Hello I am Anmol !
```

Error handling

- While making a fifo named pipe using mkfifo() :

```
if (mkfifo (my_fifo_name, 0664) == -1)
        perror ("mkfifo");
```

- While opening any named pipe using open() :

```
if ((fd_server = open (SERVER_FIFO, O_WRONLY)) == -1) {
    perror ("open: server fifo");
    break; }
```

- While writing to a named pipe using write() :
```
if (write (fd_server, newstring, strlen (newstring)) != strlen (newstring)) {
    perror ("write");
    break;
}
```

- While reading from named pipe using read() :
```
if ((bytes_read = read (fd, buf2, sizeof (buf2))) == -1)
        perror ("read");
```

- While closing a named pipe:
```
if (close (fd) == -1) {
        perror ("close");
        break;
}
```

- Joining the threads :
```
if(pthread_join(sending_thread, NULL))
      fprintf(stderr, "Error joining sending thread\n");
```

Reference Links:

- https://www.softprayog.in/programming/interprocess-communication-using-fifos-in-linux
- https://www.geeksforgeeks.org/named-pipe-fifo-example-c-program/
- https://timmurphy.org/2010/05/04/pthreads-in-c-a-minimal-working-example/
- https://www.tutorialspoint.com/inter_process_communication/inter_process_communication_named_pipes.htm