Midterm Report – CSE344

Mehmet Mahir Kayadelen
Student no. 210104004252

March 2023

1 Composition

For this project, I used a seperate c file for each side server.c and client.c and one makefile for compile run and clean them for easy testability.

2 Program Logic

The server-client program consists of two main components: the server and the client. The server is responsible for accepting client connections, managing client requests, and providing appropriate responses. The client connects to the server and sends commands, receiving responses from the server.

1)Server side

The server initializes the necessary components and sets up signal handling. It establishes communication channels with clients through named pipes, creating separate read and write pipes. The server then creates a semaphore to limit the number of simultaneous client connections to a specified maximum. The server opens a general log file and keeps all the logs there, it also opens a client-specific log file when any client comes and writes each client-specific log. The server enters a loop where it waits for client connection requests. Upon receiving a request, it checks if the maximum number of clients has been reached. If the limit is not exceeded, the server acquires the semaphore, forks a child process to handle the client, and assigns it a unique process ID. The child process handles client commands by reading them from the named pipe and executing the appropriate actions based on the command type(in handle_client_command function). The server keeps track of the child processes and their PIDs. If the maximum number of clients has been reached, the server informs the client that the server's capacity is full. The server also manages the termination of child processes, handling SIGCHLD signals, and cleaning up resources associated with terminated child processes. When the server receives a termination signal, such as SIGUSR1 or SIGINT, it gracefully shuts down by terminating all child processes, disconnecting clients, releasing resources, and exiting the program.

Problems and Solutions)

Race conditions:

-Server uses flock when reading and writing files(mutual exclusion)

-Server creates a semaphore to limit the number of simultaneous client connections to a specified maximum

Signal handlings:

The handle_signal function is responsible for handling SIGUSR1 and SIGINT signals. If a SIGINT signal is received, the server prints a termination message and proceeds to terminate all child processes and connected clients and cleans resources then terminates. If a SIGUSR1 signal is received (possibly due to a client's "killServer" command), the server sends a termination signal to the parent process (itself) to initiate the shutdown process.

The handle_sigchld function is responsible for handling the SIGCHLD signal, which is triggered when a child process terminates. The server waits for child processes to terminate using waitpid and retrieves the termination status. When a child process terminates increments the value of the semaphore by 1, its PID is removed from the child_pids array.

2)Client side

The client code establishes a connection with the server by opening a named pipe for writing and optionally for reading. It sends a connection request to the server and waits for the server's response. The client then enters a command handling loop where it prompts the user for commands, processes them accordingly, and communicates with the server by writing to the named pipe. The client supports commands such as listing files, reading file content, writing to files, uploading and downloading files, quitting, and killing the server. It waits for the server's responses for certain commands and displays them to the user. The client also handles termination requests gracefully by implementing signal handling. It captures signals such as SIGUSR2, which indicates the server is shutting down, and SIGINT, which allows the client to exit gracefully by notifying the server. Additionally, the client prevents the use of SIGTSTP, as it is not allowed for client-side operations. When receiving these signals, the client takes appropriate actions such as displaying messages, sending requests to the server, and closing the named pipes before exiting.

3 Testing

list,readF,writeT,upload,download,quit,killServer

Test 1) A single client connects to the server, performs basic operations(operation for a large file (>100 MB)), and disconnects successfully.

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ls
Here      Makefile-2:Zone.Identifier  client    server
Makefile  TelegramToUpload.exe        client.c  server.c
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$
```

1

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm/Here$ ls
TelegramToDownload.exe  toRead.txt
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm/Here$
```

2

(1&2) Before performing operations Working Directory and Server's Directory

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ./server Here 3
>> Server Started PID 26881
>> Waiting for clients...
>>Client PID:'26885' connected as 'client1'
client1 disconnected
^C>>Received signal 2, terminating...
>>bye
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$
```

3

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ./client Connect 26881
Waiting for server queue...Connection established.
>>Enter command: list
Files in server directory:
TelegramToDownload.exe
all_logs.txt
log_26885.txt
toRead.txt
>>Enter command: readF toRead.txt
File content:
Hello from the
        toRead.txt file :)

>>Enter command: writeT toRead.txt Hello:)
Write successful.
>>Enter command: readF toRead.txt
File content:
Hello from the
        toRead.txt file :)
Hello:)

>>Enter command: download TelegramToDownload.exe
Server response:
File downloaded successfully. Transferred 132896656 bytes.
>>Enter command: upload TelegramToUpload.exe
Server response:
File uploaded successfully. Transferred 132896656 bytes.
>>Enter command: list
Files in server directory:
TelegramToDownload.exe
all_logs.txt
log_26885.txt
toRead.txt
TelegramToUpload.exe
>>Enter command: quit
quit request sent to server.
Log message from server: client is shutting down.
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$
```

4

(3&4) Connect and Performing operations Working Directory and Server's Directory

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ls -l
total 259680
drwx------ 2 mahir mahir      4096 May 16 21:36 Here
-rw-r--r-- 1 mahir mahir        88 May 16 13:11 Makefile
-rw-r--r-- 1 mahir mahir        27 May 16 13:11 Makefile-2:Zone.Identifier
-rw-r--r-- 1 mahir mahir 132896656 May 16 21:36 TelegramToDownload.exe
-rw-r--r-- 1 mahir mahir 132896656 Apr 25 11:58 TelegramToUpload.exe
-rwxr-xr-x 1 mahir mahir     22048 May 16 21:21 client
-rw-r--r-- 1 mahir mahir     12551 May 16 14:36 client.c
-rwxr-xr-x 1 mahir mahir     31792 May 16 21:21 server
-rw-r--r-- 1 mahir mahir     23642 May 16 21:12 server.c
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$
```

5

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm/Here$ ls -l
total 259580
-rw-r--r-- 1 mahir mahir 132896656 Apr 25 11:58 TelegramToDownload.exe
-rw-r--r-- 1 mahir mahir 132896656 May 16 21:36 TelegramToUpload.exe
-rw------- 1 mahir mahir       400 May 16 21:37 all_logs.txt
-rw------- 1 mahir mahir       135 May 16 21:36 log_26885.txt
-rw-r--r-- 1 mahir mahir        46 May 16 21:35 toRead.txt
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm/Here$
```

6

(5&6) After performing operations Working Directory and Server's Directory

We can see that the upload and download operations are successful.

Test2) Multiple clients try to connect to the server, but the server's client capacity is full. When space is free, it connects and sends killServer command. Server shuts down all clients

In order to understand the input/output behavior in different SS, I have colored it, each color represents a certain time.

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ./server dirname 2
>> Server Started PID 27522
>> Waiting for clients...
>>Client PID:'27524' connected as 'client1'
>>Client PID:'27533' connected as 'client2'
>>Connection request PID 27540… Que FULL. Client leaving...
>>Connection request PID 27541… Que FULL.
client1 disconnected
>>Client PID:'27541' connected as 'client3'
```

1

First Connection:

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ./client Connect 27522
Waiting for server queue...Connection established.
>>Enter command: quit
quit request sent to server.
Log message from server: client is shutting down.
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$
```

2

Second Connection:

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ./client tryConnect 27522
Connection established.
>>Enter command:
```

3

Third Connection attempt with tryConnect(no slot) after with Connect option(waits queue till first connection quits.):

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ./client tryConnect 27522
Server is Full!
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ./client Connect 27522
Waiting for server queue...Connection established.
>>Enter command:
```

4

Sending killServer from last client:

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ./client tryConnect 27522
Server is Full!
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ./client Connect 27522
Waiting for server queue...Connection established.
>>Enter command: killServer
Kill request sent to server.
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$
```

5

Server terminates other clients and childs, After terminates itself

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ./server dirname 2
>> Server Started PID 27522
>> Waiting for clients...
>>Client PID:'27524' connected as 'client1'
>>Client PID:'27533' connected as 'client2'
>>Connection request PID 27540… Que FULL. Client leaving...
>>Connection request PID 27541… Que FULL.
client1 disconnected
>>Client PID:'27541' connected as 'client3'
>>Kill signal from client3.. Terminating...
>>bye
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$
```

6

Client killed by server:

```
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$ ./client tryConnect 27522
Connection established.
>>Enter command: Killed
mahir@DESKTOP-9RRECEV:~/CSE344-SYSTEM/Midterm$
```

7