

Distributed Operating System Principles

Programming Assignment 1

REPORT

TEAM-27

NAME	UFID	EMAIL
Rhith Sai Reddy Kondreddy	65682267	r.kondreddy@ufl.edu
Guna Teja Athota	36434173	gunateja.athota@ufl.edu
Karnam Sushmanth	46615547	Karnam.sushmanth@ufl.edu
Obellaneni Satya Aakash Chowdary	26077180	s.obellaneni@ufl.edu

Contents:

- 1. Project Description**
 - a. Objectives**
 - b. Requirements**
- 2. Compilation and Execution of the code**
- 3. Code Structure Description**
- 4. Results of Execution**
- 5. Conclusion**
- 6. Individual Contributors**

1. Project Description

Objectives:

This report details the concurrent client/server socket communication system's implementation, and testing in F#. The task entails creating several client applications, a server program, and a socket-based TCP/IP protocol suite for communication between the programs. The report provides information on the system's general performance as well as the implementation process, problems encountered, and remedies developed.

Requirements:

1. Server- Side Script:

- We built our server application to accommodate several client connections at once. The server enters a listening state after initialization and awaits incoming client requests. The server starts asynchronous tasks as soon as a connection is made to handle each client separately. The overall performance of the system is improved by this concurrent method.

2. Client- Side Script:

- The client software connects to the server and communicates with the user. It displays the responses after sending asynchronous user commands to the server. By processing termination commands from both the client and server sides, graceful termination is ensured.

3. Exception Handling:

- Include robust exception management, including the production and processing of error codes, to guarantee dependable system operation. Make sure termination methods are tidy to prevent resource leaks and runaway processes.

4. Termination:

- Termination in a client/server model refers to the process of controlling and gracefully terminating the connection and communication between clients and the server. To guarantee that resources are released, connections are closed, and the system departs without leaving any lingering processes or open sockets, proper termination is crucial. The client/server model can implement termination as follows:

2. Compilation and Execution of the code

- Create a directory for the project.
- To build a server folder, enter the command **dotnet new console --language F# -o server** in the project directory.
- To create a client folder, enter the command **dotnet new console --language F# -o client** in the project directory.
- Rename the files Program.fs, client.fs, and server.fs in the client directory and server directory, respectively.
- Additionally, modify the filename of fsproj files.
- Both the server.fsproj file should compile server.fs file and the client.fsproj file should compile the client.fs file.
- To compile the project, launch a terminal, go to the server directory (cd server), and then type **dotnet build**.
- Launch another terminal, go to the client directory, and run the **dotnet build** command to compile the project.
- Execute the dotnet run command later in the server terminal to start the server. Run the client application by launching the client terminal and entering the dotnet run command.
- Open as many client terminals as necessary, then repeat the process of using the dotnet run command as you did for the first client. Every client terminal functions as a separate client that you can use to provide instructions to the server and get replies from it.
- When a new client-server connection is made with the help of the dotnet run command, wait for the server to send the client a "Hello!" message.

- After that, clients can ask the server for things and the server can answer them.

3. Code Structure Description

Each client communication is handled in its own thread in the code, which is designed to handle numerous clients concurrently. It exhibits suitable error management, response generation, and termination techniques, making it appropriate for an F# socket communication project.

Client-Side code:

1. `handleClient(client: TcpClient)` Function:

Purpose: asynchronously manages communication with a particular client.

Components:

- Message from the client is read, handled using `handleCommand`, and then the client is sent the response.
- after processing, disconnects the client connection.

Code Organization and Flow:

Modularity: Functions are organized based on their specific tasks, promoting modularity and ease of maintenance.

Asynchronous Processing: Asynchronous programming is utilized to handle multiple clients concurrently, ensuring efficient use of server resources.

Error Handling: The code includes comprehensive error handling, addressing various exceptions and providing appropriate error messages.

Graceful Termination: The server can be gracefully shut down using the "terminate" command, closing all connections and threads.

4. Results of Execution

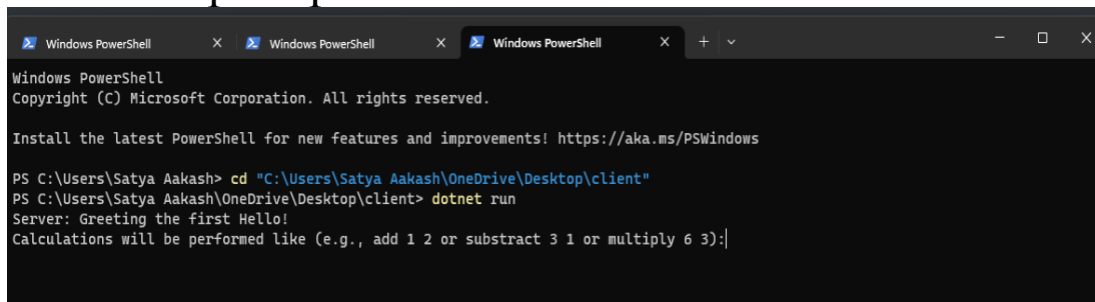
First server will listen to the port 8080 before connecting to the client server will look like this

```
C:\Users\Satya Aakash\OneDrive\Desktop\server>dotnet run
C:\Users\Satya Aakash\OneDrive\Desktop\server\Program.fs(60,21): warning FS0020: The result of this expression has type 'string * int array' and is implicitly ignored. Consider using 'ignore' to discard this value explicitly, e.g. 'expr |> ignore', or 'let' to bind the result to a name, e.g. 'let result = expr'
. [C:\Users\Satya Aakash\OneDrive\Desktop\server\server.fsproj]
server is hosted on 0.0.0.0
Server is listening on port 8080
```

When we start the client we will get the greeting message hello from the client

```
PS C:\Users\Satya Aakash> cd "C:\Users\Satya Aakash\OneDrive\Desktop\client"
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Encountered socket issue: No connection could be made because the target machine actively refused it. [::ffff:192.168.0.223]:8080
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Server: Greeting the first Hello!
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):|
```

When we connect multiple clients to the server the client command prompt is

A screenshot of a Windows PowerShell window with three tabs. The active tab shows the following text:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Satya Aakash> cd "C:\Users\Satya Aakash\OneDrive\Desktop\client"
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Server: Greeting the first Hello!
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):|
```

When multiple clients are connected to the server the command prompt of server is

```
C:\Users\Satya Aakash\OneDrive\Desktop\server\Program.fs(60,21): warning FS0020: The result of this expression has type 'string * int array' and is implicitly ignored. Consider using 'ignore' to discard this value explicitly, e.g. 'expr |> ignore', or 'let' to bind the result to a name, e.g. 'let result = expr'
. [C:\Users\Satya Aakash\OneDrive\Desktop\server\server.fsproj]
server is hosted on 0.0.0.0
Server is listening on port 8080
The client 1 is connected...
The client 2 is connected...
The client 3 is connected...
```

Test case:

- Client 1: Sent a command add 4 5 to the server.
- Got result 9 from the server

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Satya Aakash> cd "C:\Users\Satya Aakash\OneDrive\Desktop\client"
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Encountered socket issue: No connection could be made because the target machine actively refused it. [::ffff:192.168.0.223]:8080
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Server: Greeting the first Hello!
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Server: Greeting the first Hello!
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):add 4 5
Server: 9
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):|
```

Server side

```
C:\Users\Satya Aakash\OneDrive\Desktop\server>dotnet run
C:\Users\Satya Aakash\OneDrive\Desktop\server\Program.fs(60,21): warning FS0020: The result of this expression has type 'string * int array' and is implicitly ignored. Consider using 'ignore' to discard this value explicitly, e.g. 'expr |> ignore', or 'let' to bind the result to a name, e.g. 'let result = expr'
. [C:\Users\Satya Aakash\OneDrive\Desktop\server\server.fsproj]
server is hosted on 0.0.0.0
Server is listening on port 8080
The client 1 is connected...
The client 2 is connected...
The client 3 is connected...
Received the command: add 4 5
Responding to client 1 with result: 9
|
```

Test case: Add 3 4 5 6

Sent a command add 3 4 5 6 to the server

Received a result 18 from the server

```
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Server: Greeting the first Hello!
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Server: Greeting the first Hello!
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):add 4 5
Server: 9
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):add 3 4 5 6
Server: 18
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):|
```

Server side

```
value explicitly, e.g. 'expr |> ignore', or 'let' to bind the result to a name, e.g. 'let result = expr'
. [C:\Users\Satya Aakash\OneDrive\Desktop\server\server.fsproj]
server is hosted on 0.0.0.0
Server is listening on port 8080
The client 1 is connected...
The client 2 is connected...
The client 3 is connected...
Received the command: add 4 5
Responding to client 1 with result: 9
Received the command: add 3 4 5 6
Responding to client 1 with result: 18
```

Test case: Subtract 13 1

Sent a command subtract 13 1 to the server

Received 12 from the server

```
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
server: Greeting the first Hello!
calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):subtract 13 1
server: 12
calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):
```

Server side

```
server is hosted on 0.0.0.0
Server is listening on port 8080
The client 1 is connected...
Received the command: subtract 13 1
Responding to client 1 with result: 12
```

Test case: Multiply 6 7

Sent a command Multiply 6 7 to the server

Received 42 from the server

```
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
server: Greeting the first Hello!
calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):multiply 6 7
server: 42
calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):
```


Server side

```
C:\Users\Satya Aakash\OneDrive\Desktop\server>dotnet run
server is hosted on 0.0.0.0
Server is listening on port 8080
The client 1 is connected...
Received the command: multiply 6 7
Responding to client 1 with result: 42
```

Exception Handling

Test case 1: An 1 4 (Incorrect operation command)

Send a command An 1 4 to the server

Received -1 from the server

```
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Server: Greeting the first Hello!
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):an 1 4
Server: -1
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):|
```

Server side

```
C:\Users\Satya Aakash\OneDrive\Desktop\server>dotnet run
server is hosted on 0.0.0.0
Server is listening on port 8080
The client 1 is connected...
Received the command: an 1 4
Responding to client 1 with result: -1
```

Test case 2: add 1 (number of inputs is less than two)

Send a command Add 1 to the server

Received -2 from the server

```
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Server: Greeting the first Hello!
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):an 1 4
Server: -1
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):add 1
Server: -2
```

Server side

```
Received the command: add 1  
Responding to client 1 with result: -2
```

Test case 3: add 1 2 3 4 5 (number of inputs is more than four)

Send a command add 1 2 3 4 5 to the server

Received -3 from the server

```
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run  
Server: Greeting the first Hello!  
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):an 1 4  
Server: -1  
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):add 1  
Server: -2  
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):add 1 2 3 4 5  
Server: -3
```

Server side

```
server is hosted on 0.0.0.0  
Server is listening on port 8080  
The client 1 is connected...  
Received the command: an 1 4  
Responding to client 1 with result: -1  
Received the command: add 1  
Responding to client 1 with result: -2  
Received the command: add 1 2 3 4 5  
Responding to client 1 with result: -3
```

Test case 4: add a b (one or more of the inputs contains non numbers)

Send a command add a b to the server

Received -4 from the server

```
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):add a b  
Server: -4
```

Server side

```
Received the command: add a b  
Responding to client 1 with result: -4  
|
```

Test case 5: Bye (exit)

Send a command bye to the server

Received exit from the server

```
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):bye
Terminating connection.
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> |
```

Server side

```
Client 1 disconnected
Client 2 disconnected
|
```

Performing addition operation on 3 clients

```
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):add 9 10
Server: 19
```

```
PS C:\Users\Satya Aakash> cd "C:\Users\Satya Aakash\OneDrive\Desktop\client"
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Server: Greeting the first Hello!
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):add 9 18
Server: 27
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):|
```

```
PS C:\Users\Satya Aakash> cd "C:\Users\Satya Aakash\OneDrive\Desktop\client"
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> dotnet run
Server: Greeting the first Hello!
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):add 16 9
Server: 25
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):|
```

Server side

```
Received the command: add 9 10
Responding to client 1 with result: 19
The client 2 is connected...
Received the command: add 9 18
Responding to client 2 with result: 27
The client 3 is connected...
Received the command: add 16 9
Responding to client 3 with result: 25
|
```

Terminating the connection Using Terminate command

```
Calculations will be performed like (e.g., add 1 2 or subtract 3 1 or multiply 6 3):terminate
Terminating connection.
PS C:\Users\Satya Aakash\OneDrive\Desktop\client> |
```

Server side

```
C:\Users\Satya Aakash\OneDrive\Desktop\server>  
C:\Users\Satya Aakash\OneDrive\Desktop\server>|
```

5. Conclusion

Concurrent socket communication is provided by this project using F# programming. The project focuses heavily on reliable and error-resistant answers, strong exception handling, and termination techniques in order to build a server that can efficiently manage multiple client connections. This project also introduces the concept of asynchronous client-server communication, allowing clients to talk to the server at the same time.

6. Individual Contributions

a. Guna Teja Athota:

- Implemented the server program in F#
- Implemented handling of concurrent client request
- Ensured that server can manage several clients all at once
- Implemented termination logic depending on requests of clients
- Implemented client initialization and making sure of connection to server
- Implemented server initialization , listening for connections and accepting them
- Implemented the TCP/IP protocol suite and configuring socket communication

b. Kondreddy Rohith Sai Reddy

- Implemented the F# client program
- Established TCP/IP socket contact with the server.
- Implemented server connection and client initialization
- Implemented the logic to calculate and carry out commands based on input from the client.
- Acquire server answers and print them.
- Implemented basic error handling for client inputs
- For various error codes, print the associated error messages.

c. Karnam Sushmanth

- Improved the client to handle command input from the user.
- Created a loop so users can engage continuously (enter commands, submit them to the server, and then receive responses).
- Implemented a Logic for analyzing server answers, including error codes, should be included.
- For various error codes, print the associated error messages.
- When the server returns a #-5 response, implement graceful termination for the client.

d. Aakash

- Created test cases for various circumstances, including valid and invalid commands, multiple clients, and termination situations.
- Concurrency and effective client-server communication have been tested throughout the entire system.
- Keep a record of the testing procedure, including the inputs, anticipated results, and actual results.
- Write a report that includes team information, compilation and execution instructions, a description of the code structure, and an analysis of the execution results.
- Included screenshots in the report to show that the implementation was successful.