COP5615- Distributed Operating System Principles

Fall 2023

Programming Assignment #1

Team 27

Group members:

Name	UFID	Email
Rishika Reddy Alugubelli	**2888-0035**	**r.alugubelli@ufl.edu**
Aashritha Reddy Donapati	**7242-2083**	**a.donapati@ufl.edu**
Vivek Reddy Gangula	**5208-7488**	**gangula.v@ufl.edu**
Dheekshita Neella	**6325-8153**	**d.neella@ufl.edu**

Table of Contents

1. Compilation, Execution of the Code and its Environment:

- Editor Used : Visual Studio Code
  - Programming Language: F#
  - Supported OS: Windows, MacOS
- Install Ionide for F# Plugin for Visual Studio Code
- Install .NET SDK from dotnet.microsoft.com/download
- Download the PA1_Team27.zip file and extract it.
- Open the program.fs files under server and client folders respectively in VSCode.
- In the client file, change the IP address in the 8th line to the IP Address of the system that is being used to run the code.
- Open 2 or more terminals in VSCode and navigate to the server folder in one terminal and client folder in the rest of the terminals.
- For compilation, run the command “dotnet build” in all the terminals. If the compilation is successful, it displays ‘Build Success’.
- For execution, run the command “dotnet run”, first in the server terminal then in the client terminals. This will start an interactive session between the server and clients.

2. Code Structure:

The code is in 2 files, server file and client file

2.1 Server:

handleClient Function:

- The handleClient function handles the communication with each client asynchronously.
- Increments a variable “clientCounter” for each new client connection
- A stream, reader and writer are initialized for communicating with the client
- As we establish a connection, it sends a welcome message
- It handles commands such as "bye" to close the connection for a client and "terminate" to close all connections and exit.
- It also performs arithmetic operations like addition, subtraction, multiplication, and division when the given command is valid.
- It contains a switch case which handles various errors such as the number of inputs less than two or greater than four, and invalid inputs.
- It also takes care of the exceptions like IOException and FormatException.

Main Function:

- Creates a TcpListener that listens for incoming connections.
- A message server is listening on the specified port is printed.
- The acceptClients function is used to handle the asynchronous acceptance of new clients.
- It accepts clients asynchronously.
- It accepts new clients until isTerminated is true.

Error Handling:

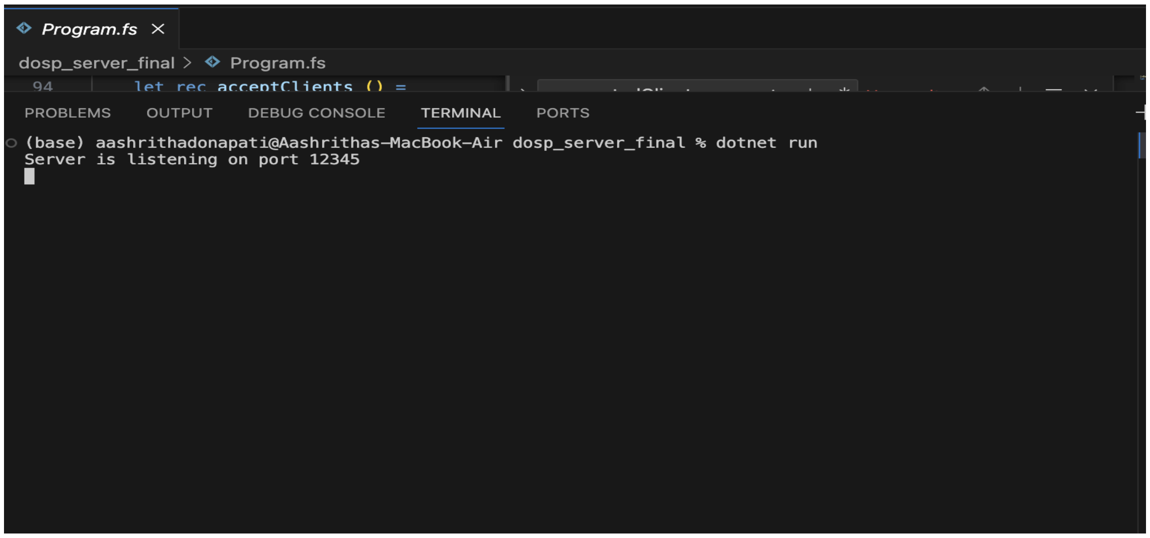
- The code includes exception handling for various scenarios, such as IO errors, client disconnects, and other exceptions.

2.2 Client:

- The client program connects to a server, sends commands, receives responses and outputs different error messages based on the error codes received.
- Necessary namespaces are imported and the IP address and port number of the server have been stored in the variable(Note: serverPort should be replaced with the server's port number and the IP address should be replaced with the server's IP address).
- asyncConnectAndCommunicate () handles the connection to the server.
- The try block establishes a connection, reads and writes to the server, and handles SocketException and IOException.
- It creates a TcpClient, connects to the server, and sets up StreamReader and StreamWriter for reading and writing to the server stream.
- The receiving welcome message reads the server's message and prints it to the console.
- There's a loop for sending commands to the server and receiving responses which terminates when we receive error code -5.
- User input prompts to enter a command, reads the input and sends the command to the server.
- Response Handling: Reads the server's response, matches it against error codes and prints corresponding messages. If the server responds with -5(exit code), it prints an exit message and terminates the client.
- The main function runs the asyncconnectandcommunicate() function and returns an exit code.

3. Execution Results and their Explanation:

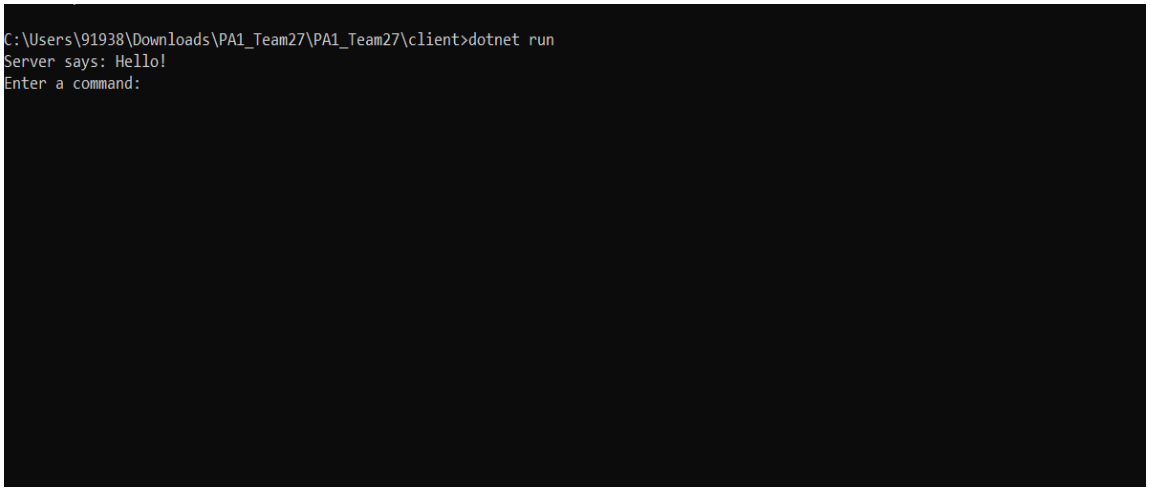
1. **Establishing a connection**
    - Server



```
let rec acceptClients () =
```

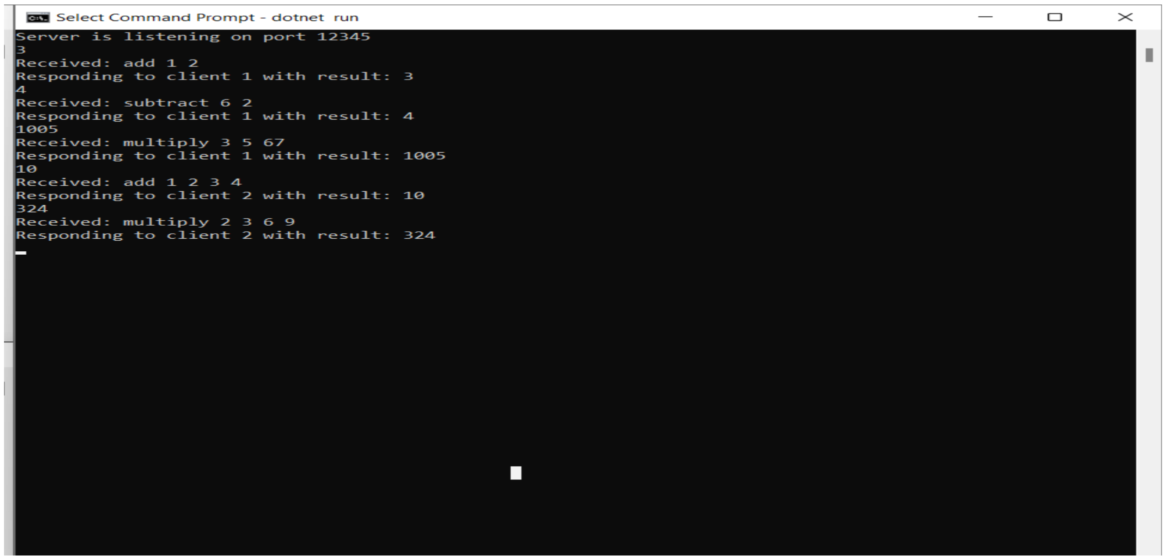
```
(base) aashrithadonapati@Aashrithas-MacBook-Air dosp_server_final % dotnet run
Server is listening on port 12345
```

    - Client



```
C:\Users\91938\Downloads\PA1_Team27\PA1_Team27\client>dotnet run
Server says: Hello!
Enter a command:
```

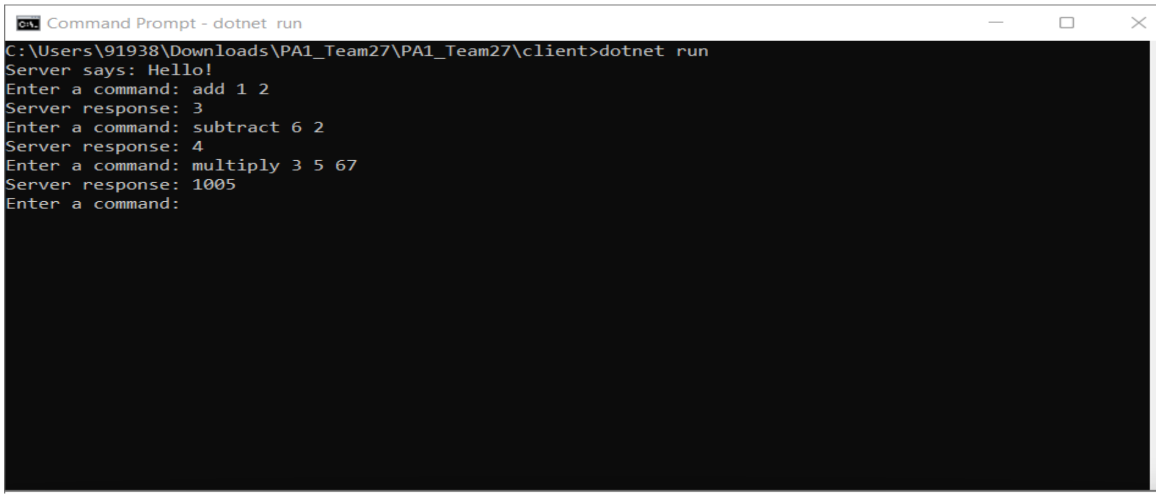
2. **Add, Subtract and Multiply Operations**
    - server



```
Select Command Prompt - dotnet  run
Server is listening on port 12345
3
Received: add 1 2
Responding to client 1 with result: 3
4
Received: subtract 6 2
Responding to client 1 with result: 4
1005
Received: multiply 3 5 67
Responding to client 1 with result: 1005
10
Received: add 1 2 3 4
Responding to client 2 with result: 10
324
Received: multiply 2 3 6 9
Responding to client 2 with result: 324
```

Server Terminal: Here It receives the commands sent by the client, performs operations specified and then sends a response to the client. After executing client1 requests, the server accepts requests from multiple clients asynchronously.

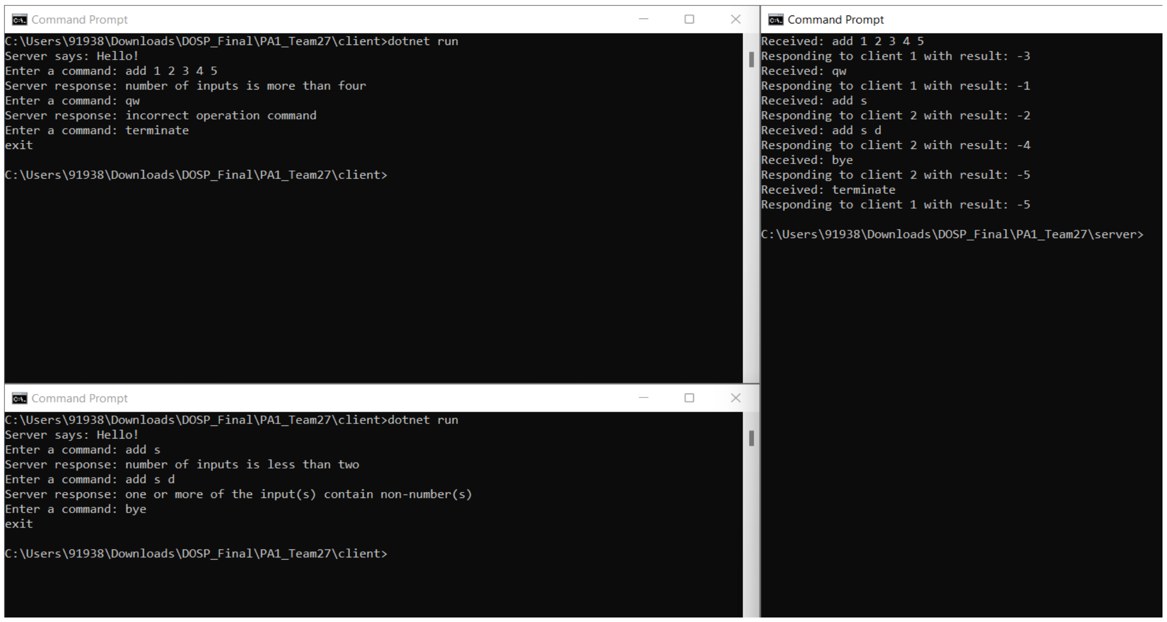
- Client



```
Command Prompt - dotnet  run
C:\Users\91938\Downloads\PA1_Team27\PA1_Team27\client>dotnet run
Server says: Hello!
Enter a command: add 1 2
Server response: 3
Enter a command: subtract 6 2
Server response: 4
Enter a command: multiply 3 5 67
Server response: 1005
Enter a command:
```

Client 1 terminal: Here, Hello! Message displayed indicates the connection establishment between client and server. We've given commands to add, subtract, and multiply. When we enter a command like "subtract 6 2," the server performs the subtraction on the given numbers and sends the response back as 4.

3. **Exception Handling:**

:

```
Command Prompt
C:\Users\91938\Downloads\DOSP_Final\PA1_Team27\client>dotnet run
Server says: Hello!
Enter a command: add 1 2 3 4 5
Server response: number of inputs is more than four
Enter a command: qw
Server response: incorrect operation command
Enter a command: terminate
exit

C:\Users\91938\Downloads\DOSP_Final\PA1_Team27\client>
```

```
Command Prompt
C:\Users\91938\Downloads\DOSP_Final\PA1_Team27\client>dotnet run
Server says: Hello!
Enter a command: add s
Server response: number of inputs is less than two
Enter a command: add s d
Server response: one or more of the input(s) contain non-number(s)
Enter a command: bye
exit

C:\Users\91938\Downloads\DOSP_Final\PA1_Team27\client>
```

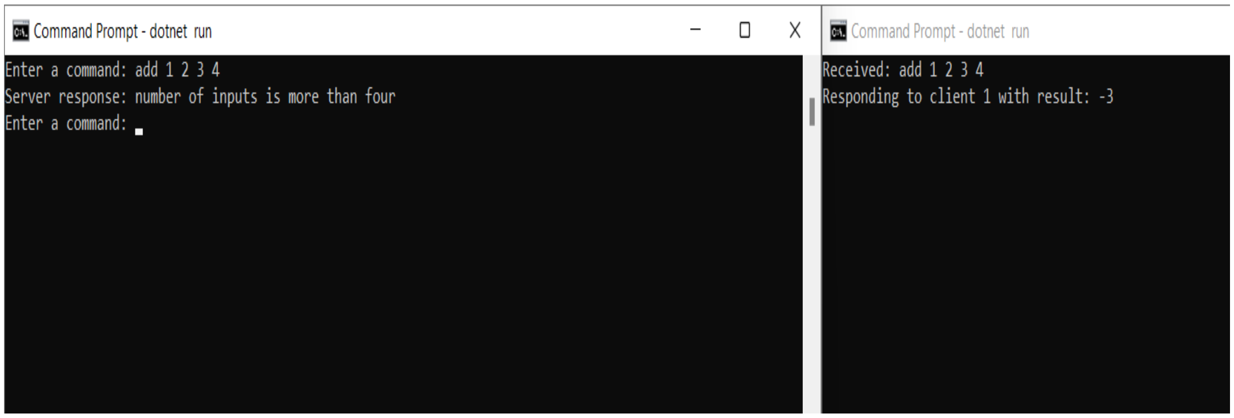
```
Command Prompt
Received: add 1 2 3 4 5
Responding to client 1 with result: -3
Received: qw
Responding to client 1 with result: -1
Received: add s
Responding to client 2 with result: -2
Received: add s d
Responding to client 2 with result: -4
Received: bye
Responding to client 2 with result: -5
Received: terminate
Responding to client 1 with result: -5

C:\Users\91938\Downloads\DOSP_Final\PA1_Team27\server>
```

Server Terminal: Here, it prints the command request received by the client and the response sent along with the clientCounter. We will display the error code in the server terminal, whereas in the client terminal, we will print the respective error message.

Client 1 Output Terminal, Here the client sends an input command as add 1 2 3 4 5, the server sends an error message “number of inputs is more than four” as response, then the client sends a command “qw” which is not an operation, the server sends a response as “incorrect operation command”, and when the serve enters a command “terminate” the server exits and all the clients gets terminated.

Client 2 Output Terminal, Here, the client gives a single input and non numeric input, the server sends an error message as “no of inputs is less than two” whose code is lesser than “One or more of the inputs contain(s) non-number(s)”, then the client enters non numeric inputs, therefore the server sends a response as “One or more of the inputs contain(s) non-number(s)” and then the client gives command “bye”, the server responses with exit and client terminates.

4. Abnormal Results:



```
Enter a command: add 1 2 3 4
Server response: number of inputs is more than four
Enter a command:
```

```
Received: add 1 2 3 4
Responding to client 1 with result: -3
```

Here, the client gives the command as “add 1 2 3 4 “ along with an empty space. Although there are only 4 characters, an empty space is also included and the server sends a response as -3 i.e “number of inputs is more than four” error message, as it considers space as an additional input.

5. Team members contribution:

Rishika Reddy Alugubelli:

- Referred F# tutorial tutorials, installation guides for project setup.
- Implemented client side logic for connecting to the server.
- Tested bugs in the code
- Worked on programming assignment report.

Aashritha Reddy Donapati:

- Implemented client side logic for connecting to the server.
- Tested the program under multiple scenarios.
- Worked on programming assignment report.
- Code Review

Vivek Reddy Gangula:

- Implemented the server-side logic for handling connections, requests and responses.
- Integrated server and client components.
- Implemented error handling mechanism on server side program.
- Tested the program under multiple scenarios.

Dheekshita Neella:

- Implemented the server-side logic for handling connections, requests and responses.
- Integrated server and client components.
- Implemented error handling mechanism on server side program.
- Tested bugs in the code.