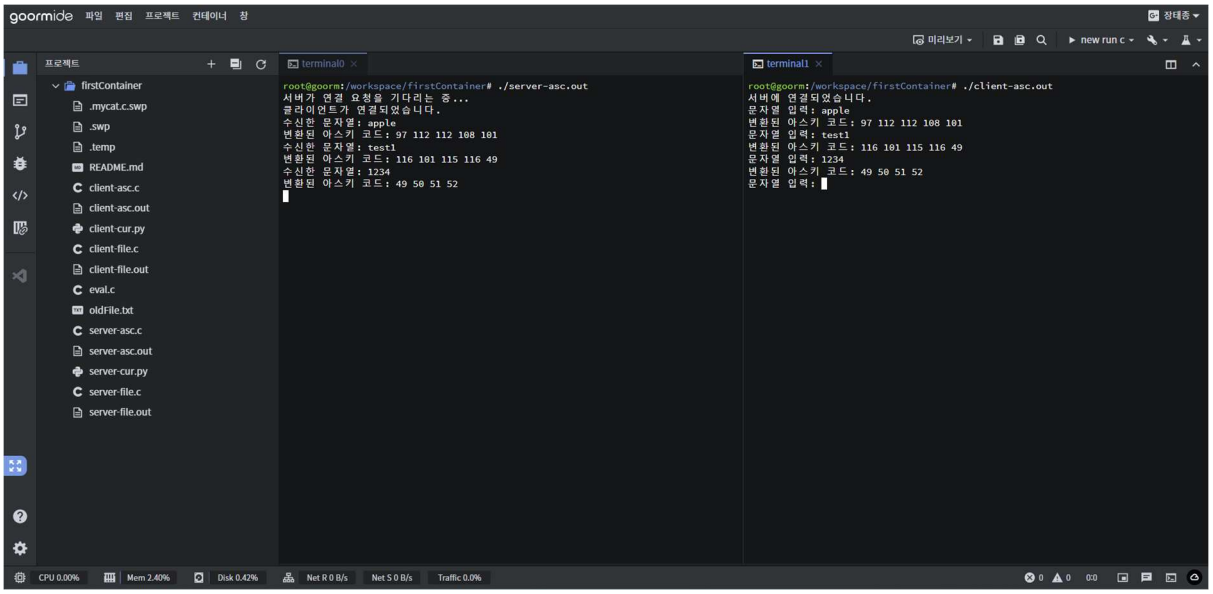
1. 클라이언트로부터 문자열을 입력받아 아스키코드로 변환하여 반환하는 소켓프로그래밍



서버1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>

#define PORT 8080
#define MAX_BUFFER_SIZE 1024

int main() {
    int serverSocket, newSocket;
    struct sockaddr_in serverAddr, clientAddr;
    char buffer[MAX_BUFFER_SIZE];

    // 소켓 생성
    serverSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (serverSocket == -1) {
        perror("소켓 생성 실패");
        exit(EXIT_FAILURE);
    }

    // 서버 주소 설정
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    serverAddr.sin_port = htons(PORT);

    // 소켓 바인딩
    if (bind(serverSocket, (struct sockaddr*)&serverAddr, size
        perror("바인딩 실패");
        exit(EXIT_FAILURE);
    }

    // 클라이언트의 연결 요청 대기
    if (listen(serverSocket, 5) == -1) {
        perror("리스닝 실패");
        exit(EXIT_FAILURE);
    }
```

서버2

```
37
38      printf("서버가 연결 요청을 기다리는 중...\n");
39
40      // 클라이언트 연결 수락
41      socklen_t clientAddrLen = sizeof(clientAddr);
42      newSocket = accept(serverSocket, (struct sockaddr*)&client
43      if (newSocket == -1) {
44          perror("연결 수락 실패");
45          exit(EXIT_FAILURE);
46      }
47
48      printf("클라이언트가 연결되었습니다.\n");
49
50      // 클라이언트로부터 데이터 수신 및 변환 후 전송
51      while (1) {
52          memset(buffer, 0, sizeof(buffer));
53
54          // 클라이언트로부터 데이터 수신
55          if (recv(newSocket, buffer, sizeof(buffer), 0) == -1)
56              perror("수신 실패");
57              exit(EXIT_FAILURE);
58          }
59
60          if (strlen(buffer) == 0) {
61              printf("클라이언트가 연결을 종료했습니다.\n");
62              break;
63          }
64
65          printf("수신한 문자열: %s\n", buffer);
66
67          // 아스키 코드로 변환
68          char asciiBuffer[MAX_BUFFER_SIZE * 3];  // 변환된 아스키
69          memset(asciiBuffer, 0, sizeof(asciiBuffer));
70
71          for (int i = 0; i < strlen(buffer); i++) {
72              sprintf(asciiBuffer + strlen(asciiBuffer), "%d ",
73          }
74
75          printf("변환된 아스키 코드: %s\n", asciiBuffer);
76
```

서버3

```
77          // 변환된 아스키 코드 전송
78          if (send(newSocket, asciiBuffer, strlen(asciiBuffer),
79              perror("전송 실패");
80              exit(EXIT_FAILURE);
81          }
82      }
83
84      // 소켓 닫기
85      close(newSocket);
86      close(serverSocket);
87
88      return 0;
89  }
90
```

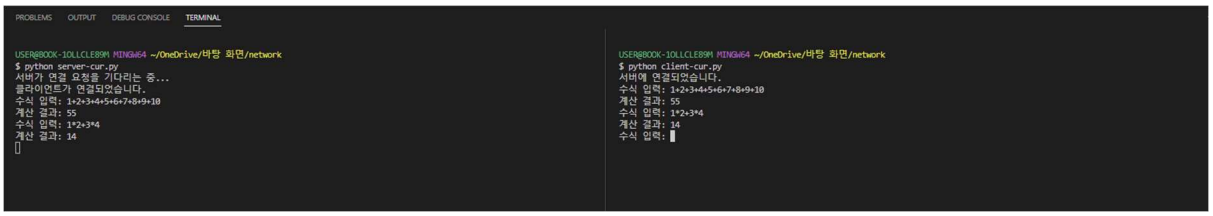
클라1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>

#define SERVER_IP "127.0.0.1"
#define PORT 8080
#define MAX_BUFFER_SIZE 1024

int main() {
    int clientSocket;
    struct sockaddr_in serverAddr;
    char buffer[MAX_BUFFER_SIZE];

    // 소켓 생성
    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (clientSocket == -1) {
        perror("소켓 생성 실패");
        exit(EXIT_FAILURE);
    }

    // 서버 주소 설정
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);

    // 서버 IP 주소 설정
    if (inet_pton(AF_INET, SERVER_IP, &(serverAddr.sin_addr))
        perror("IP 주소 설정 실패");
        exit(EXIT_FAILURE);
    }

    // 서버에 연결
    if (connect(clientSocket, (struct sockaddr*)&serverAddr, s
        perror("연결 실패");
        exit(EXIT_FAILURE);
    }

    printf("서버에 연결되었습니다.\n");

```

클라2

```c
    // 사용자 입력 받고 서버로 전송
    while (1) {
        printf("문자열 입력: ");
        fgets(buffer, sizeof(buffer), stdin);
        buffer[strcspn(buffer, "\n")] = '\0';  // 개행 문자 제거

        if (strlen(buffer) == 0) {
            printf("연결을 종료합니다.\n");
            break;
        }

        // 서버로 데이터 전송
        if (send(clientSocket, buffer, strlen(buffer), 0) == -
            perror("전송 실패");
            exit(EXIT_FAILURE);
        }

        memset(buffer, 0, sizeof(buffer));

        // 서버로부터 데이터 수신
        if (recv(clientSocket, buffer, sizeof(buffer), 0) == -
            perror("수신 실패");
            exit(EXIT_FAILURE);
        }

        printf("변환된 아스키 코드: %s\n", buffer);
    }

    // 소켓 닫기
    close(clientSocket);

    return 0;
}
```

2. 클라이언트로부터 숫자를 수식을 입력받아 계산한 결과를 반환하는 소켓프로그래밍



서버 1

```python
import socket
import math

SERVER_IP = '127.0.0.1'
PORT = 8080
MAX_BUFFER_SIZE = 1024


def calculate_expression(expression):
    try:
        result = eval(expression)
        return str(result)
    except Exception as e:
        print("수식 계산 중 오류 발생:", e)
        return "Error"


def main():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_
    server_socket.bind((SERVER_IP, PORT))
    server_socket.listen(1)

    print("서버가 연결 요청을 기다리는 중...")

    client_socket, client_addr = server_socket.accept()
    print("클라이언트가 연결되었습니다.")

    while True:
        data = client_socket.recv(MAX_BUFFER_SIZE).decode()

        if not data:
            print("클라이언트가 연결을 종료했습니다.")
            break

        print("수식 입력:", data)
        result = calculate_expression(data)
        print("계산 결과:", result)

        client_socket.send(result.encode())
```

서버 2

```
40
41      client_socket.close()
42      server_socket.close()
43
44
45  if __name__ == '__main__':
46      main()
47
```

클라 1

```
1   import socket
2
3   SERVER_IP = '127.0.0.1'
4   PORT = 8080
5   MAX_BUFFER_SIZE = 1024
6
7
8   def main():
9       client_socket = socket.socket(socket.AF_INET, socket.SOCK_
10      client_socket.connect((SERVER_IP, PORT))
11
12      print("서버에 연결되었습니다.")
13
14      while True:
15          expression = input("수식 입력: ")
16
17          if not expression:
18              print("연결을 종료합니다.")
19              break
20
21          client_socket.send(expression.encode())
22
23          result = client_socket.recv(MAX_BUFFER_SIZE).decode()
24          print("계산 결과:", result)
25
26      client_socket.close()
27
28
29  if __name__ == '__main__':
30      main()
31
```

3. 클라이언트로부터 요청받은 파일명에 해당하는 내용을 반환하는 소켓프로그래밍

```
root@goorm:/workspace/firstContainer# ./server-file.out
Server is listening on port 12345...
root@goorm:/workspace/firstContainer#
```

```
root@goorm:/workspace/firstContainer# ./client-file.out
Enter the file name: eval.c
#include <Python.h>

int main(void)
{
        Py_Initialize();
        if (Py_IsInitialized())
        {
                printf("success");
        }
        return 0;
}
root@goorm:/workspace/firstContainer#
```

서버1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define PORT 12345
#define BUFFER_SIZE 1024

int main() {
    int serverSocket, clientSocket;
    struct sockaddr_in serverAddr, clientAddr;
    socklen_t addrLen = sizeof(clientAddr);
    char buffer[BUFFER_SIZE];

    // 소켓 생성
    if ((serverSocket = socket(AF_INET, SOCK_STREAM, 0)) == -1
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // 서버 주소 설정
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    memset(serverAddr.sin_zero, '\0', sizeof(serverAddr.sin_ze

    // 소켓과 서버 주소 바인딩
    if (bind(serverSocket, (struct sockaddr *)&serverAddr, siz
        perror("Binding failed");
        exit(EXIT_FAILURE);
    }

    // 클라이언트로부터 연결 요청 대기
    if (listen(serverSocket, 1) == -1) {
        perror("Listening failed");
        exit(EXIT_FAILURE);
    }

```

```
    printf("Server is listening on port %d...\n", PORT);

    // 클라이언트 연결 수락
    if ((clientSocket = accept(serverSocket, (struct sockaddr
        perror("Accepting failed");
        exit(EXIT_FAILURE);
    }

    // 클라이언트로부터 파일명 수신
    memset(buffer, 0, BUFFER_SIZE);
    if (recv(clientSocket, buffer, BUFFER_SIZE, 0) == -1) {
        perror("Receiving failed");
        exit(EXIT_FAILURE);
    }

    // 파일 열기
    FILE *file = fopen(buffer, "r");
    if (file == NULL) {
        perror("File opening failed");
        exit(EXIT_FAILURE);
    }

    // 파일 내용 읽기
    memset(buffer, 0, BUFFER_SIZE);
    while (fgets(buffer, BUFFER_SIZE, file) != NULL) {
        // 클라이언트로 파일 내용 전송
        if (send(clientSocket, buffer, strlen(buffer), 0) == -
            perror("Sending failed");
            exit(EXIT_FAILURE);
        }
        memset(buffer, 0, BUFFER_SIZE);
    }

    // 파일 닫기
    fclose(file);

    // 클라이언트 소켓 종료
    close(clientSocket);
```

서버2

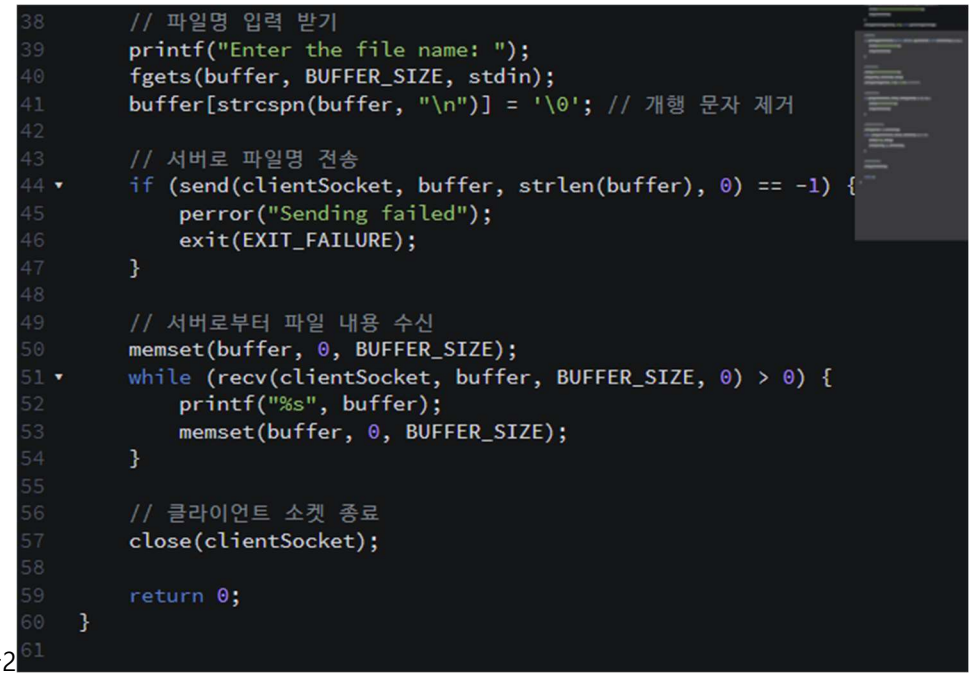
```
    // 서버 소켓 종료
    close(serverSocket);

    return 0;
}
```

서버3

클라1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define SERVER_IP "127.0.0.1"
#define PORT 12345
#define BUFFER_SIZE 1024

int main() {
    int clientSocket;
    struct sockaddr_in serverAddr;
    char buffer[BUFFER_SIZE];

    // 소켓 생성
    if ((clientSocket = socket(AF_INET, SOCK_STREAM, 0)) == -1
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // 서버 주소 설정
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);
    if (inet_pton(AF_INET, SERVER_IP, &(serverAddr.sin_addr))
        perror("Invalid address/ Address not supported");
        exit(EXIT_FAILURE);
    }
    memset(serverAddr.sin_zero, '\0', sizeof(serverAddr.sin_ze

    // 서버에 연결
    if (connect(clientSocket, (struct sockaddr *)&serverAddr,
        perror("Connection failed");
        exit(EXIT_FAILURE);
    }
```

클라2



```c
    // 파일명 입력 받기
    printf("Enter the file name: ");
    fgets(buffer, BUFFER_SIZE, stdin);
    buffer[strcspn(buffer, "\n")] = '\0'; // 개행 문자 제거

    // 서버로 파일명 전송
    if (send(clientSocket, buffer, strlen(buffer), 0) == -1) {
        perror("Sending failed");
        exit(EXIT_FAILURE);
    }

    // 서버로부터 파일 내용 수신
    memset(buffer, 0, BUFFER_SIZE);
    while (recv(clientSocket, buffer, BUFFER_SIZE, 0) > 0) {
        printf("%s", buffer);
        memset(buffer, 0, BUFFER_SIZE);
    }

    // 클라이언트 소켓 종료
    close(clientSocket);

    return 0;
}
```