# Assignment2

**김성록(KIM SEONGROK)**

**2016116783**

# Example1

## 1-1. Echo_server.c

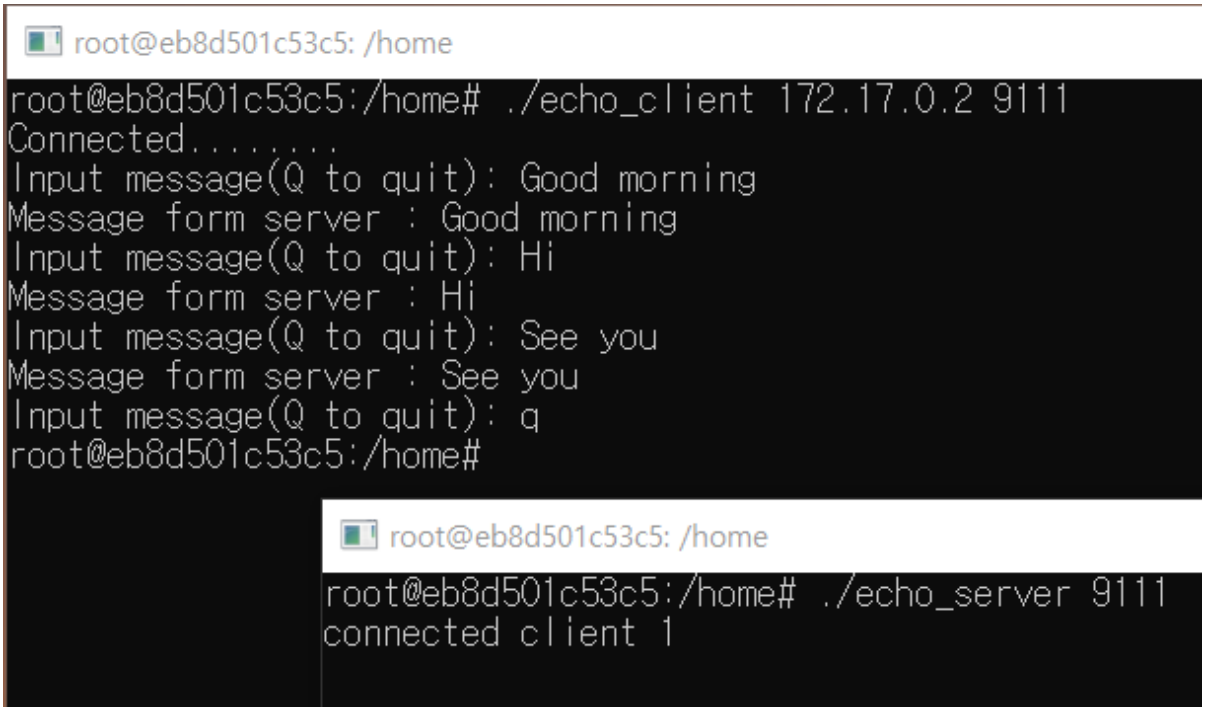
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 1024

void error_handling(char *message);

int main(int argc, char* argv[]){
  int serv_sock;
  int clnt_sock;

  struct sockaddr_in serv_addr, clnt_addr;
  socklen_t clnt_addr_size;
  char message[BUF_SIZE];
  int str_len, i;
  if(argc != 2){
          printf("Usage: %s <port>\n", argv[0]);
          exit(1);
  }
  serv_sock= socket(PF_INET, SOCK_STREAM, 0);
  if(serv_sock==-1)
          error_handling("socket() error");

  memset(&serv_addr, 0, sizeof(serv_addr));
  serv_addr.sin_family = AF_INET;
  serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
  serv_addr.sin_port = htons(atoi(argv[1]));
  if(bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
    error_handling("bind() error!");
  if(listen(serv_sock, 5)==-1)
    error_handling("listen() error!");
  clnt_addr_size = sizeof(clnt_addr);

  for(i=0; i<5; i++){
    clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_addr, &clnt_addr_size);
    if(clnt_sock==-1)
      error_handling("accept() error");
    else
      printf("connected client %d \n", i+1);
    while((str_len = read(clnt_sock, message, BUF_SIZE)) != 0)
      write(clnt_sock, message, str_len);
    close(clnt_sock);
  }
  close(serv_sock);
  return 0;
}
```

```
void error_handling(char *message){
        fputs(message, stderr);
        fputc('\n',stderr);
        exit(1);
 }
```

## 1-2. Echo.client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 1024

void error_handling(char *message);

int main(int argc, char* argv[]){
        int sock;
        struct sockaddr_in serv_addr;
        char message[BUF_SIZE];
        int str_len;
        if(argc != 3){
                printf("Usage: %s <IP> <port>\n", argv[0]);
        }
        sock = socket(PF_INET, SOCK_STREAM, 0);
        if(sock==-1)
                error_handling("socket() error");

        memset(&serv_addr, 0, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
        serv_addr.sin_port = htons(atoi(argv[2]));

        if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
                error_handling("connect() error!");
        else
                puts("Connected........");

        while(1){
                fputs("Input message(Q to quit): ", stdout);
                fgets(message, BUF_SIZE, stdin);

                if(!strcmp(message, "q\n") || !strcmp(message, "Q\n"))
                        break;
                write(sock, message, strlen(message));
                str_len = read(sock, message, BUF_SIZE-1);
                message[str_len] = 0;
                printf("Message form server : %s", message);
        }
        close(sock);
        return 0;
}



void error_handling(char *message){
        fputs(message, stderr);
        fputc('\n',stderr);
```

```
        exit(1);
 }
```




# Example2

## 2-1. op_server.c

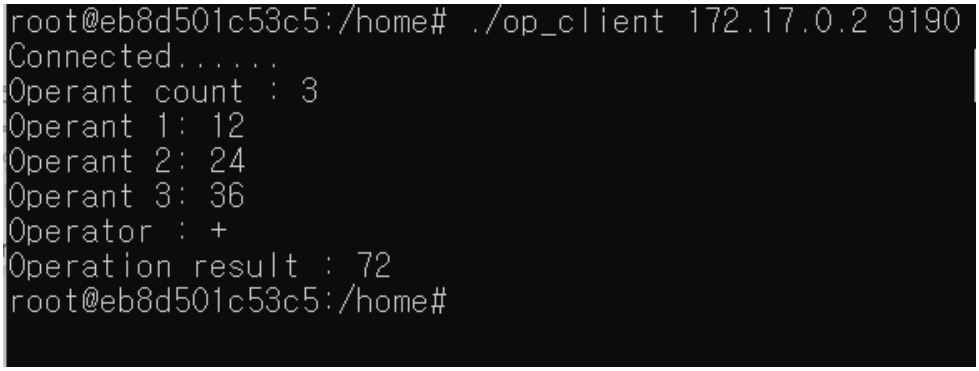
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 1024
#define OPSZ 4

void error_handling(char *message);
int calculate(int opsum, int opnds[], char operator);

int main(int argc, char *argv[]){
        int serv_sock, clnt_sock;
        char opinfo[BUF_SIZE];
        int result, opnd_cnt, i;
        int recv_cnt, recv_len;

        struct sockaddr_in serv_adr, clnt_adr;
        socklen_t clnt_adr_sz;
        if(argc!=2){
                printf("Usage : %s <port>\n", argv[0]);
                exit(1);
        }
        serv_sock = socket(PF_INET, SOCK_STREAM, 0);
```

```
        if(serv_sock==-1)
                error_handling("socket() error!");
        memset(&serv_adr, 0, sizeof(serv_adr));
        serv_adr.sin_family = AF_INET;
        serv_adr.sin_addr.s_addr = htonl(INADDR_ANY);
        serv_adr.sin_port = htons(atoi(argv[1]));
        if(bind(serv_sock, (struct sockaddr*)&serv_adr, sizeof(serv_adr))==-1)
                error_handling("bind() error!");
        if(listen(serv_sock, 5==-1))
                error_handling("listen() error!");
        clnt_adr_sz = sizeof(clnt_adr);
        for(i=0;i<5;i++){
                opnd_cnt=0;
                clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_adr, &clnt_adr_sz);
                read(clnt_sock, &opnd_cnt, 1);
                recv_len = 0;
                while((opnd_cnt * OPSZ + 1) > recv_len){
                        recv_cnt = read(clnt_sock, &opinfo[recv_len], BUF_SIZE-1);
                        recv_len += recv_cnt;
                }
                result = calculate(opnd_cnt, (int*)opinfo, opinfo[recv_len-1]);
                write(clnt_sock, (char*)&result, sizeof(result));
                close(clnt_sock);
        }
        close(serv_sock);
        return 0;
}

void error_handling(char *message){
        fputs(message, stderr);
        fputc('\n',stderr);
        exit(1);
}

int calculate(int opnum, int opnds[], char op){
        int result=opnds[0],i;
        switch(op){
                case '+' :
                        for(i=1;i<opnum;i++)
                                result+=opnds[i];
                        break;
                case '-' :
                        for(i=1;i<opnum;i++)
                                result-=opnds[i];
                        break;
                case '*' :
                        for(i=1;i<opnum;i++)
                                result*=opnds[i];
                        break;
        }
        return result;
}
```

## 2-2. op_client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 1024
#define OPSZ 4
#define RLT_SIZE 4
```

```
void error_handling(char *message);

int main(int argc, char *argv[]){
        int sock;
        char opmsg[BUF_SIZE];
        int result, opnd_cnt, i;
        struct sockaddr_in serv_adr;
        if(argc!=3){
                printf("Usage : %s <IP> <port>\n", argv[0]);
                exit(1);
        }
        sock = socket(PF_INET, SOCK_STREAM, 0);
        if(sock==-1)
                error_handling("socket() error!");
        memset(&serv_adr, 0, sizeof(serv_adr));
        serv_adr.sin_family = AF_INET;
        serv_adr.sin_addr.s_addr = inet_addr(argv[1]);
        serv_adr.sin_port = htons(atoi(argv[2]));

        if(connect(sock, (struct sockaddr*)&serv_adr, sizeof(serv_adr))==-1)
                error_handling("connect() error!");
        else
                puts("Connected......");

        fputs("Operant count : ", stdout);
        scanf("%d", &opnd_cnt);
        opmsg[0] = (char)opnd_cnt;
        for(i=0;i<opnd_cnt;i++){
                printf("Operant %d: ", i+1);
                scanf("%d", (int*)&opmsg[i*OPSZ+1]);
        }
        fgetc(stdin);
        fputs("Operator : ", stdout);
        scanf("%c", &opmsg[opnd_cnt * OPSZ+1]);
        write(sock, opmsg, opnd_cnt * OPSZ+2);
        read(sock, &result, RLT_SIZE);

        printf("Operation result : %d\n", result);
        close(sock);
        return 0;
}

void error_handling(char *message){
        fputs(message, stderr);
        fputc('\n',stderr);
        exit(1);
}
```




# Example3

## 3-1. uecho_server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 1024

void error_handling(char *message);

int main(int argc, char *argv[])
{
    int serv_sock;
    char message[BUF_SIZE];
    int str_len;
    socklen_t clnt_addr_size;

    struct sockaddr_in serv_addr, clnt_addr;
    if (argc != 2)
    {
        printf("Usage: %s <port>\n", argv[0]);
        exit(1);
    }
    serv_sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (serv_sock == -1)
        error_handling("UDP socker creation error");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(atoi(argv[1]));

    if (bind(serv_sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) == -1)
        error_handling("bind() error!");


    while(1){
        clnt_addr_size = sizeof(clnt_addr);
        str_len = recvfrom(serv_sock, message, BUF_SIZE, 0, (struct sockaddr*)&clnt_addr, &clnt_addr_size);
        sendto(serv_sock, message, str_len, 0, (struct sockaddr*)&clnt_addr, clnt_addr_size);
```

```
    }
    close(serv_sock);
    return 0;
}

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```

## 3-2. uecho_client.c

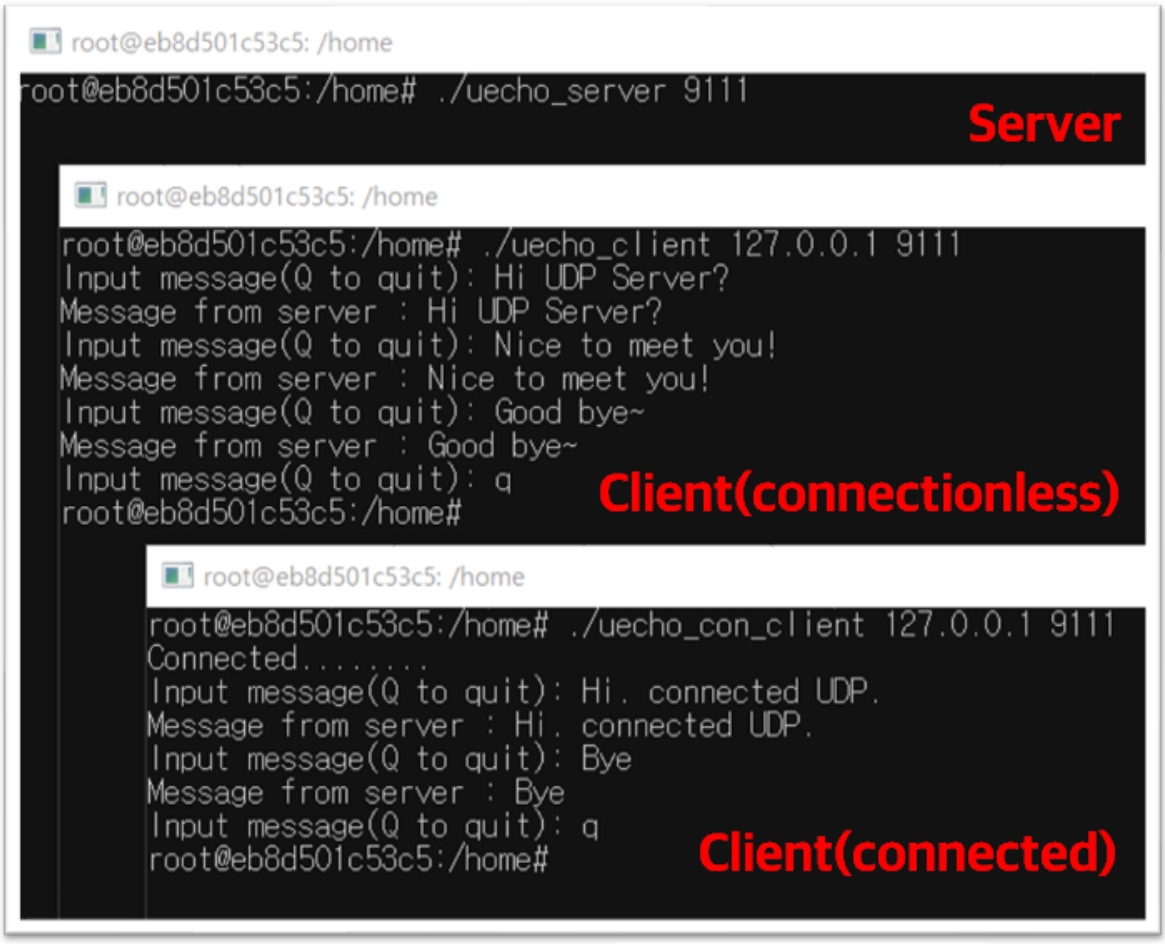
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30

void error_handling(char *message);

int main(int argc, char *argv[])
{
    int sock;
    char message[BUF_SIZE];
    int str_len;
    socklen_t addr_sz;
    struct sockaddr_in serv_addr, from_addr;

    if (argc != 3)
    {
        printf("Usage: %s <IP> <port>\n", argv[0]);
        exit(1);
    }

    sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (sock == -1)
        error_handling("socket() error");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_addr.sin_port = htons(atoi(argv[2]));

    while (1)
    {
        fputs("Input message(Q to quit): ", stdout);
        fgets(message, BUF_SIZE, stdin);

        if (!strcmp(message, "q\n") || !strcmp(message, "Q\n"))
            break;
        sendto(sock, message, strlen(message), 0, (struct sockaddr*)&serv_addr, sizeof(serv_addr));;
        addr_sz = sizeof(from_addr);
        str_len = recvfrom(sock, message, BUF_SIZE, 0, (struct sockaddr*)&from_addr, &addr_sz);
        message[str_len]=0;
        printf("Message from server : %s", message);
    }
    close(sock);
    return 0;
}

void error_handling(char *message)
```

```
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```

## 3-3. uecho_con_client.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30

void error_handling(char *message);

int main(int argc, char *argv[])
{
    int sock;
    char message[BUF_SIZE];
    int str_len;
    socklen_t addr_sz;
    struct sockaddr_in serv_addr, from_addr;

    if (argc != 3)
    {
        printf("Usage: %s <IP> <port>\n", argv[0]);
        exit(1);
    }

    sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (sock == -1)
        error_handling("socket() error");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_addr.sin_port = htons(atoi(argv[2]));

    if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
                error_handling("connect() error!");
    else
            puts("Connected........");

    while (1)
    {
        fputs("Input message(Q to quit): ", stdout);
        fgets(message, BUF_SIZE, stdin);

        if (!strcmp(message, "q\n") || !strcmp(message, "Q\n"))
            break;

        write(sock, message, strlen(message));
        str_len = read(sock, message, sizeof(message)-1);
        message[str_len]=0;
        printf("Message from server : %s", message);
    }
    close(sock);
    return 0;
}

void error_handling(char *message)
{
```

```
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```




# Example 4

## 4-1. bound_host1.c

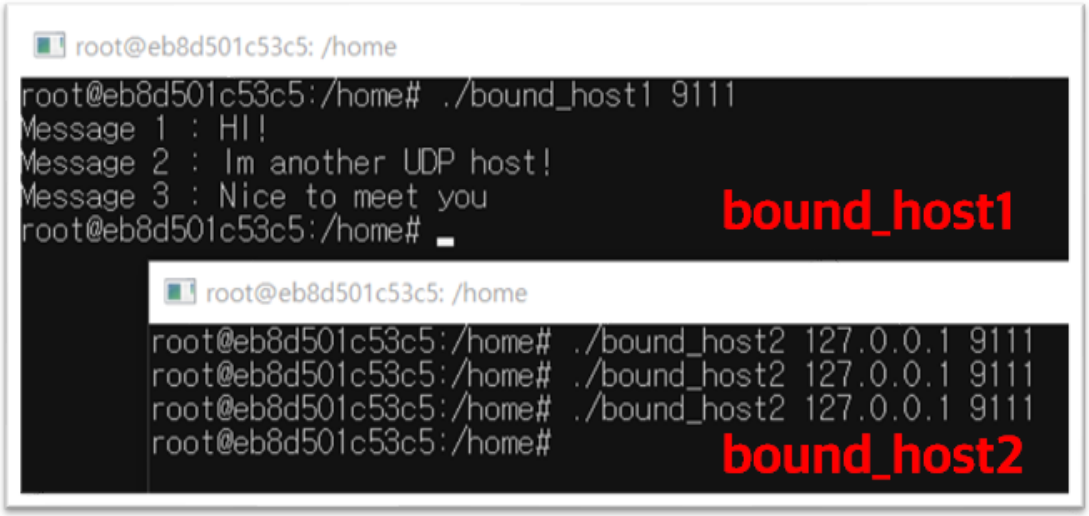
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30

void error_handling(char *message);

int main(int argc, char *argv[])
```

```
{
    int sock;
    char message[BUF_SIZE];

    struct sockaddr_in my_addr, your_addr;
    socklen_t addr_size;

    int str_len, i;
    if (argc != 2)
    {
        printf("Usage: %s <port>\n", argv[0]);
        exit(1);
    }
    sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (sock == -1)
        error_handling("socket() error");

    memset(&my_addr, 0, sizeof(my_addr));
    my_addr.sin_family = AF_INET;
    my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    my_addr.sin_port = htons(atoi(argv[1]));

    if (bind(sock, (struct sockaddr *)&my_addr, sizeof(my_addr)) == -1)
        error_handling("bind() error!");

    for (i = 0; i < 3; i++)
    {
        sleep(5);
        addr_size = sizeof(your_addr);
        str_len = recvfrom(sock, message, BUF_SIZE, 0,
                           (struct sockaddr *)&your_addr, &addr_size);
        printf("Message %d : %s\n", i + 1, message);
    }
    close(sock);
    return 0;
}

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```

## 4-2. bound_host2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30

void error_handling(char *message);

int main(int argc, char *argv[])
{
    int sock;
    char msg1[] = "HI!";
    char msg2[] = "Im another UDP host!";
    char msg3[] = "Nice to meet you";

    struct sockaddr_in your_addr;
    socklen_t addr_size;
```

```
    int str_len, i;
    if (argc != 3)
    {
        printf("Usage: %s <IP> <port>\n", argv[0]);
        exit(1);
    }
    sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (sock == -1)
        error_handling("socket() error");

    memset(&your_addr, 0, sizeof(your_addr));
    your_addr.sin_family = AF_INET;
    your_addr.sin_addr.s_addr = inet_addr(argv[1]);
    your_addr.sin_port = htons(atoi(argv[2]));

    sendto(sock, msg1, sizeof(msg1), 0,
           (struct sockaddr *)&your_addr, sizeof(your_addr));
    sendto(sock, msg2, sizeof(msg2), 0,
           (struct sockaddr *)&your_addr, sizeof(your_addr));
    sendto(sock, msg3, sizeof(msg3), 0,
           (struct sockaddr *)&your_addr, sizeof(your_addr));

    close(sock);
    return 0;
}

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```




# Problem

## op_server_iter.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <math.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 1024
#define OPSZ 4

void error_handling(char *message);
int calculate(int opnum, int opnds[], char op);

int main(int argc, char *argv[])
{
    int serv_sock;
    char message[BUF_SIZE];
    int str_len;
    socklen_t clnt_addr_size;
    char opinfo[BUF_SIZE];
    int result, opnd_cnt, i;
    int recv_cnt, recv_len;
    struct sockaddr_in serv_addr, clnt_addr;
    if (argc != 2)
    {
        printf("Usage: %s <port>\n", argv[0]);
        exit(1);
    }
    serv_sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (serv_sock == -1)
        error_handling("UDP socker creation error");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(atoi(argv[1]));

    if (bind(serv_sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) == -1)
        error_handling("bind() error!");

    while (1)
    {
        opnd_cnt = 0;
        clnt_addr_size = sizeof(clnt_addr);

        str_len = recvfrom(serv_sock, opinfo, BUF_SIZE, 0, (struct sockaddr*)&clnt_addr, &clnt_addr_size);
        opnd_cnt = opinfo[0];
        recv_len = 0;
        for(i=1;i<opnd_cnt*OPSZ+1;i=i+4){
            printf("%d\n",opinfo[i]);
        }
        result = calculate(opnd_cnt, (int *)(opinfo+1), opinfo[str_len - 1]);
        sendto(serv_sock, (char *)&result, OPSZ, 0, (struct sockaddr*)&clnt_addr, clnt_addr_size);
    }
    close(serv_sock);
    return 0;
}

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}

int calculate(int opnum, int opnds[], char op)
{
```

```
    int result = opnds[0], i;
    switch (op)
    {
    case '+':
        for (i = 1; i < opnum; i++)
            result += opnds[i];
        break;
    case '-':
        for (i = 1; i < opnum; i++)
            result -= opnds[i];
        break;
    case '*':
        for (i = 1; i < opnum; i++)
            result *= opnds[i];
        break;
    case '^':
        for (i = 1; i < opnum; i++)
            result = pow(result,opnds[i]);
        break;
    }
    return result;
}
```

## op_client_iter2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define BUF_SIZE 30
#define OPSZ 4
#define RLT_SIZE 4

void error_handling(char *message);

int main(int argc, char *argv[])
{
    int sock;
    char opmsg[BUF_SIZE];
    int result, opnd_cnt, i;
    int str_len;
    socklen_t addr_sz;
    struct sockaddr_in serv_addr, from_addr;

    if (argc != 3)
    {
        printf("Usage: %s <IP> <port>\n", argv[0]);
        exit(1);
    }

    sock = socket(PF_INET, SOCK_DGRAM, 0);
    if (sock == -1)
        error_handling("socket() error");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_addr.sin_port = htons(atoi(argv[2]));

    if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
                error_handling("connect() error!");
    else
            puts("Connected........");
```

```
    while (1)
    {
        fputs("Operant count (-1 to quit) : ", stdout);
        scanf("%d", &opnd_cnt);

        if (opnd_cnt==-1)
            break;

        opmsg[0] = (char)opnd_cnt;
        for (i = 0; i < opnd_cnt; i++)
        {
            printf("Operant %d: ", i + 1);
            scanf("%d", (int *)&opmsg[i * OPSZ + 1]);
        }
        fgetc(stdin);
        fputs("Operator : ", stdout);
        scanf("%c", &opmsg[opnd_cnt * OPSZ + 1]);

        write(sock, opmsg, opnd_cnt * OPSZ + 2);
        str_len = read(sock, &result, RLT_SIZE);
        printf("Operation result : %d\n", result);
    }
    close(sock);
    return 0;
}

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```