

Experiment 2a

server.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
#define true 1
#define false 0

int isNumber(char *str)
{
    int i;
    for (i = 0; i < strlen(str); i++)
    {
        char ch = str[i];
        if(ch < '0' || ch > '9')
        {
            return false;
        }
    }
    return true;
}

int isValidIp(char *ip_addr)
{
    int i, j, dots = 0;
    ip_addr[strlen(ip_addr) - 1] = '\\0';
    printf("%s", ip_addr);
    if (ip_addr != NULL)
    {
        char *ptr = strtok(ip_addr, ".");

        if (ptr == NULL)
        {
            return false;
        }
        while (ptr)
        {
            if (!isNumber(ptr))
            {
                return false;
            }
            else
            {

```

```

        int num = atoi(ptr);
        if(num < 0 || num > 255)
        {
            return false;
        }
        else
        {
            ptr = strtok(NULL, ".");
            dots++;
        }
    }
    return (dots == 4)? true: false;
}
return false;
}

int main()
{
    int n, cid, sid, l;
    struct sockaddr_un s, c;
    sid = socket(AF_UNIX, SOCK_STREAM, 0);
    s.sun_family = AF_UNIX;
    strcpy(s.sun_path, "FileSocket");
    unlink("FileSocket");
    bind(sid, (struct sockaddr*)&s, sizeof(s));
    listen(sid, 1);
    printf("\nServer started. Waiting for
connection.....\n");
    l = sizeof(c);
    cid = accept(sid, (struct sockaddr*)&s, &l);
    printf("Connection Accepted. Obtaining
Inputs.....\n");
    while(1)
    {
        char *ip_addr = (char *)calloc(16,
sizeof(char));
        read(cid, ip_addr, 16);
        if (strncmp(ip_addr, "end", 3) == 0)
        {
            close(cid);
            close(sid);
            break;
        }
    }
}

```

```

        int result = isValidIp(ip_addr);
        char *out = (char *)calloc(1,
sizeof(char));
        if(result == true)
        {
            out[0] = 'Y';
        }
        else
        {
            out[0] = 'N';
        }

        printf("\nSending back reply: %c\n",
out[0]);
        write(cid, out, 1);
    }
    printf("Shutting down server.....\n");
    return 0;
}

```

client.c

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/un.h>
int main()
{
    int sid, i, N;
    sid = socket(AF_UNIX, SOCK_STREAM, 0);
    struct sockaddr_un s;
    s.sun_family = AF_UNIX;
    strcpy(s.sun_path, "FileSocket");
    connect(sid, (struct sockaddr*)&s, sizeof(s));
    printf("Connected to Server.(enter end to
exit)\n");
    while(1)
    {
        printf("Enter the IPV4 Address: ");
        char *ip_addr = (char *)calloc(16,
sizeof(char));
        fgets(ip_addr, 16, stdin);
        printf("Sending IPV4 Address to be
validated.....\n");
    }
}

```

```

        write(sid, ip_addr, strlen(ip_addr)); //
writing to the file socket

        if (strncmp(ip_addr, "end", 3) == 0)
        {
            close(sid);
            break;
        }

        char server_output;
        read(sid, &server_output, 1);

        if (server_output == 'Y')
        {
            printf("Given IP is a valid IPV4
Address.\n");
        }

        else
        {
            printf("Given IP is not a valid IPV4
Address.\n");
        }

        printf("Shutting down client connection.....\n");
        return 0;
    }
}

```

The screenshot displays two terminal windows side-by-side, both running on an Intel HP Notebook. The left terminal window shows the compilation of `client.c` using `gcc` and its execution. The output indicates that the client successfully connects to the server and sends the IP address `127.0.0.1`. The right terminal window shows the compilation of `server.c` using `gcc` and its execution. The output indicates that the server starts successfully and receives the connection from the client.

```

Intel@Intel-HP-Notebook: ~/Documents/network_lab...
Intel@Intel-HP-Notebook:~/Documents/network_lab/exp_2a$ gcc client.c -o client.o && ./client.o
client.c: In function 'main':
client.c:21:3: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-function-declaration]
  21 |     write(sid, ip_addr, strlen(ip_addr)); // writing to the file socket
      |     ^~~~~~
client.c:25:4: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
  25 |     close(sid);
      |     ^~~~~~
client.c:30:3: warning: implicit declaration of function 'read'; did you mean 'fread'? [-Wimplicit-function-declaration]
  30 |     read(sid, &server_output, 1);
      |     ^~~~~~
Connected to Server.(enter end to exit)
Enter the IPV4 Address: 127.0.0.1
Sending IPV4 Address to be validated.....
Given IP is a valid IPV4 Address.
Enter the IPV4 Address:

Intel@Intel-HP-Notebook:~/Documents/network_lab/exp_2a$ gcc server.c -o server.o && ./server.o
server.c: In function 'main':
server.c:67:2: warning: implicit declaration of function 'unlink' [-Wimplicit-function-declaration]
  67 |     unlink("FileSocket");
      |     ^~~~~~
server.c:77:3: warning: implicit declaration of function 'read'; did you mean 'fread'? [-Wimplicit-function-declaration]
  77 |     read(cid, ip_addr, 16);
      |     ^~~~~~
server.c:80:4: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
  80 |     close(cid);
      |     ^~~~~~
server.c:97:3: warning: implicit declaration of function 'write'; did you mean 'fwrite'? [-Wimplicit-function-declaration]
  97 |     write(cid, out, 1);
      |     ^~~~~~
Server started. Waiting for connection....
Connection Accepted. Obtaining Inputs....
127.0.0.1
Sending back reply: Y

```

