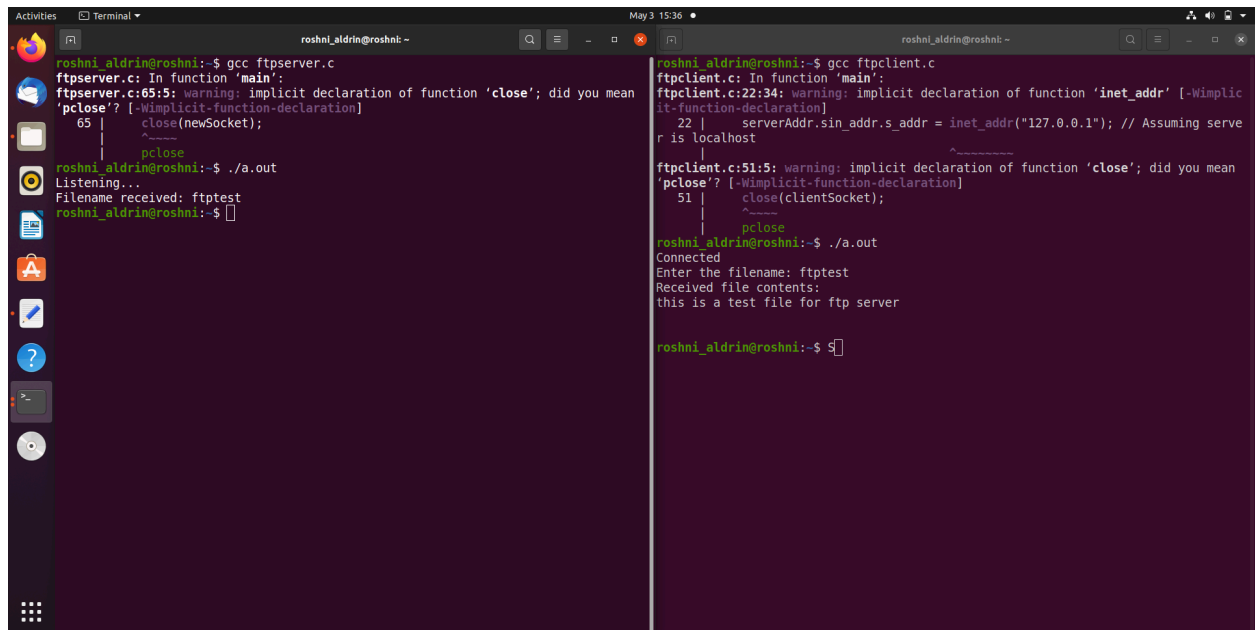


FTP SERVER



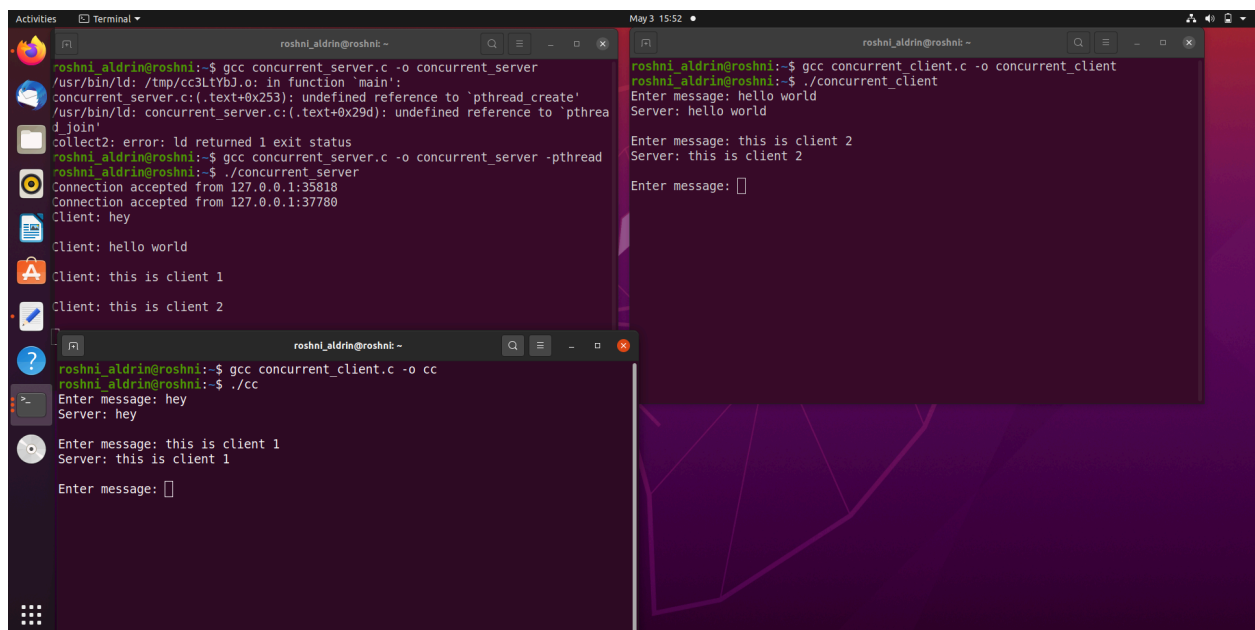
The image shows two terminal windows side-by-side. The left window shows the compilation of `ftpserver.c` with warnings about implicit declarations of `close` and `pclose`. It then shows the execution of `./a.out`, which starts listening for connections. The right window shows the compilation of `ftpclient.c` with similar warnings. It then shows the execution of `./a.out`, which connects to the server, enters the filename `ftptest`, and receives the file contents: `this is a test file for ftp server`.

```
roshni_aldrin@roshni:~$ gcc ftpserver.c
ftpserver.c: In function 'main':
ftpserver.c:65:5: warning: implicit declaration of function 'close'; did you mean
'pclose'? [-Wimplicit-function-declaration]
   65 |     close(newSocket);
      |     ^~~~~~
      |     pclose
roshni_aldrin@roshni:~$ ./a.out
Listening...
Filename received: ftptest
roshni_aldrin@roshni:~$

roshni_aldrin@roshni:~$ gcc ftpclient.c
ftpclient.c: In function 'main':
ftpclient.c:22:34: warning: implicit declaration of function 'inet_addr' [-Wimplicit-function-declaration]
   22 |     serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1"); // Assuming server
      |                                  ^~~~~~
      |                                  is localhost
ftpclient.c:51:5: warning: implicit declaration of function 'close'; did you mean
'pclose'? [-Wimplicit-function-declaration]
   51 |     close(clientSocket);
      |     ^~~~~~
      |     pclose
roshni_aldrin@roshni:~$ ./a.out
Connected
Enter the filename: ftptest
Received file contents:
this is a test file for ftp server

roshni_aldrin@roshni:~$
```

CONCURRENT SERVER



The image shows three terminal windows. The top-left window shows the compilation of `concurrent_server.c` with errors about undefined references to `pthread_create` and `pthread_join`. It then shows the execution of `./concurrent_server` with the `-pthread` flag, which starts listening for connections. The top-right window shows the compilation of `concurrent_client.c` and its execution, which sends messages to the server. The bottom window shows the execution of `./concurrent_client` with the `-cc` flag, which sends messages to the server.

```
roshni_aldrin@roshni:~$ gcc concurrent_server.c -o concurrent_server
/usr/bin/ld: /tmp/cc8LYBj.o: in function 'main':
concurrent_server.c:(.text+0x253): undefined reference to 'pthread_create'
/usr/bin/ld: concurrent_server.c:(.text+0x29d): undefined reference to 'pthread_join'
collect2: error: ld returned 1 exit status
roshni_aldrin@roshni:~$ gcc concurrent_server.c -o concurrent_server -pthread
roshni_aldrin@roshni:~$ ./concurrent_server
Connection accepted from 127.0.0.1:35818
Client: hey
Client: hello world
Client: this is client 1
Client: this is client 2

roshni_aldrin@roshni:~$ gcc concurrent_client.c -o cc
roshni_aldrin@roshni:~$ ./cc
Enter message: hey
Server: hey
Enter message: this is client 1
Server: this is client 1
Enter message:

roshni_aldrin@roshni:~$ gcc concurrent_client.c -o concurrent_client
roshni_aldrin@roshni:~$ ./concurrent_client
Enter message: hello world
Server: hello world
Enter message: this is client 2
Server: this is client 2
Enter message:
```

Expt3.5 - FTP

Client

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
int main(){
    int clientSocket;
    struct sockaddr_in serverAddr;
    socklen_t addr_size;
    clientSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (clientSocket < 0) {
        perror("Error in socket creation");
        exit(EXIT_FAILURE);
    }
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(7891); // Assuming port 7891
    serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);
    addr_size = sizeof serverAddr;
    if (connect(clientSocket, (struct sockaddr *)&serverAddr, addr_size) < 0) {
        perror("Connect failed. Error");
        exit(EXIT_FAILURE);
    }
    printf("Connected\n");
    char buffer[1024];
    printf("Enter the filename: ");
    scanf("%s", buffer);
    send(clientSocket, buffer, strlen(buffer), 0);
    char received_buffer[1024];
    int bytes_received = recv(clientSocket, received_buffer,
        sizeof(received_buffer), 0);
    if (bytes_received < 0) {
        perror("Error receiving file contents");
        exit(EXIT_FAILURE);
    }
    printf("Received file contents:\n%s\n", received_buffer);
    close(clientSocket);
    return 0;
}
```

Server

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
int main(){
    int welcomeSocket, newSocket;
    char buffer[1024];
    struct sockaddr_in serverAddr;
    struct sockaddr_storage serverStorage;
    socklen_t addr_size;
    FILE *fp;
    char str[150];
    welcomeSocket = socket(AF_INET, SOCK_STREAM, 0);
    if (welcomeSocket == -1) {
        perror("Error in socket creation");
        exit(EXIT_FAILURE);
    }
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(7891); // Assuming port 7891
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);
    if (bind(welcomeSocket, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) == -1) {
        perror("Bind failed. Error");
        exit(EXIT_FAILURE);
    }
    if (listen(welcomeSocket, 5) == 0) {
        printf("Listening...\n");
    } else {
        perror("Error in listening\n");
        exit(EXIT_FAILURE);
    }
    addr_size = sizeof(serverStorage);
    newSocket = accept(welcomeSocket, (struct sockaddr *)&serverStorage, &addr_size);
    recv(newSocket, buffer, sizeof(buffer), 0);
    printf("Filename received: %s\n", buffer);
    fp = fopen(buffer, "r");
    if (fp == NULL) {
        printf("Error opening file\n");
        exit(EXIT_FAILURE);
    }
    while (fgets(str, sizeof(str), fp) != NULL) {
        send(newSocket, str, strlen(str), 0);
    }
    fclose(fp);
    close(newSocket);
    close(welcomeSocket);
    return 0;
}
```

Expt3.6 – TCP Concurrent Server

Client

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define PORT 8080
#define SERVER_IP "127.0.0.1"

int main() {
    int client_socket;
    struct sockaddr_in server_addr;
    char buffer[1024] = {0};
    client_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (client_socket == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    if (inet_pton(AF_INET, SERVER_IP, &server_addr.sin_addr) <= 0) {
        perror("Invalid address/ Address not supported");
        exit(EXIT_FAILURE);
    }
    if (connect(client_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Connection failed");
        exit(EXIT_FAILURE);
    }
    while (1) {
        printf("Enter message: ");
        fgets(buffer, sizeof(buffer), stdin);
        send(client_socket, buffer, strlen(buffer), 0);
        memset(buffer, 0, sizeof(buffer));
        read(client_socket, buffer, sizeof(buffer));
        printf("Server: %s\n", buffer);
    }
    close(client_socket);
    return 0;
}
```

Server

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <pthread.h>
#define PORT 8080
#define MAX_CLIENTS 5

void *handle_client(void *arg) {
    int client_socket = *((int *)arg);
    char buffer[1024];
    while (1) {
        memset(buffer, 0, sizeof(buffer));
        read(client_socket, buffer, sizeof(buffer));
        printf("Client: %s\n", buffer);
        write(client_socket, buffer, strlen(buffer));
    }
}

int main() {
    int server_socket, client_socket;
    struct sockaddr_in server_addr, client_addr;
    pthread_t tid[MAX_CLIENTS];
    int client_count = 0;
    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket == -1) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }
    int opt = 1;
    if (setsockopt(server_socket, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt, sizeof(opt))) {
        perror("Setsockopt failed");
        exit(EXIT_FAILURE);
    }
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);
    if (bind(server_socket, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Bind failed");
        exit(EXIT_FAILURE);
    }
    if (listen(server_socket, 5) < 0) {
```

```
        perror("Listen failed");
        exit(EXIT_FAILURE);
    }
    while (1) {
        socklen_t client_len = sizeof(client_addr);
        client_socket = accept(server_socket, (struct sockaddr *)&client_addr, &client_len);
        if (client_socket < 0) {
            perror("Accept failed");
            exit(EXIT_FAILURE);
        }
        printf("Connection accepted from %s:%d\n", inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));
        if (pthread_create(&tid[client_count++], NULL, handle_client, (void *)&client_socket) != 0) {
            perror("Thread creation failed");
            exit(EXIT_FAILURE);
        }
        if (client_count >= MAX_CLIENTS) {
            client_count = 0;
            while (client_count < MAX_CLIENTS) {
                pthread_join(tid[client_count++], NULL);
            }
            client_count = 0;
        }
    }
    close(server_socket);
    return 0;
}
```