**Create a peer-to-peer file sharing application using IPC. Give program for this in C language.**

**Aim:**

To create a peer-to-peer file sharing application using IPC. Give program for this in C language.

**Objective:**

Creating a peer-to-peer (P2P) file sharing application using Inter-Process Communication (IPC) in C involves designing a system where multiple clients can share files with each other. Here, we can use UNIX domain sockets (for simplicity and local testing) to facilitate IPC between processes. The application will have two main components: the client and the server.

Here’s a basic implementation:

1. Server: The server will listen for incoming connections from clients and facilitate file sharing between them.
2. Client: The client will connect to the server to upload or download files.

**Program:**

**SERVER code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/un.h>

#define SOCKET\_PATH "p2p\_socket"

#define BUFFER\_SIZE 1024

void handle\_client(int client\_socket) {

char buffer[BUFFER\_SIZE];

int bytes\_read;

FILE \*file;

// Read file name

bytes\_read = read(client\_socket, buffer, BUFFER\_SIZE);

buffer[bytes\_read] = '\0';

if (strncmp(buffer, "upload ", 7) == 0) {

char \*file\_name = buffer + 7;

file = fopen(file\_name, "wb");

// Receive file content

while ((bytes\_read = read(client\_socket, buffer, BUFFER\_SIZE)) > 0) {

fwrite(buffer, 1, bytes\_read, file);

}

fclose(file);

printf("File %s received.\n", file\_name);

} else if (strncmp(buffer, "download ", 9) == 0) {

char \*file\_name = buffer + 9;

file = fopen(file\_name, "rb");

if (file) {

// Send file content

while ((bytes\_read = fread(buffer, 1, BUFFER\_SIZE, file)) > 0) {

write(client\_socket, buffer, bytes\_read);

}

fclose(file);

printf("File %s sent.\n", file\_name);

} else {

printf("File %s not found.\n", file\_name);

}

}

close(client\_socket);

}

int main() {

int server\_socket, client\_socket;

struct sockaddr\_un server\_addr;

// Create socket

server\_socket = socket(AF\_UNIX, SOCK\_STREAM, 0);

if (server\_socket == -1) {

perror("socket");

exit(EXIT\_FAILURE);

}

// Bind socket

memset(&server\_addr, 0, sizeof(struct sockaddr\_un));

server\_addr.sun\_family = AF\_UNIX;

strncpy(server\_addr.sun\_path, SOCKET\_PATH, sizeof(server\_addr.sun\_path) - 1);

unlink(SOCKET\_PATH);

if (bind(server\_socket, (struct sockaddr \*)&server\_addr, sizeof(struct sockaddr\_un)) == -1) {

perror("bind");

exit(EXIT\_FAILURE);

}

// Listen for connections

if (listen(server\_socket, 5) == -1) {

perror("listen");

exit(EXIT\_FAILURE);

}

printf("Server is listening...\n");

while (1) {

// Accept connection

client\_socket = accept(server\_socket, NULL, NULL);

if (client\_socket == -1) {

perror("accept");

exit(EXIT\_FAILURE);

}

// Handle client in a new process

if (fork() == 0) {

close(server\_socket);

handle\_client(client\_socket);

exit(EXIT\_SUCCESS);

} else {

close(client\_socket);

}

}

close(server\_socket);

unlink(SOCKET\_PATH);

return 0;

}  
  
  
**CLIENT code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/un.h>

#define SOCKET\_PATH "p2p\_socket"

#define BUFFER\_SIZE 1024

void upload\_file(int socket, const char \*file\_name) {

char buffer[BUFFER\_SIZE];

FILE \*file;

// Send command

snprintf(buffer, sizeof(buffer), "upload %s", file\_name);

write(socket, buffer, strlen(buffer));

// Send file content

file = fopen(file\_name, "rb");

if (file) {

while (!feof(file)) {

int bytes\_read = fread(buffer, 1, BUFFER\_SIZE, file);

write(socket, buffer, bytes\_read);

}

fclose(file);

printf("File %s uploaded.\n", file\_name);

} else {

perror("fopen");

}

}

void download\_file(int socket, const char \*file\_name) {

char buffer[BUFFER\_SIZE];

FILE \*file;

int bytes\_read;

// Send command

snprintf(buffer, sizeof(buffer), "download %s", file\_name);

write(socket, buffer, strlen(buffer));

// Receive file content

file = fopen(file\_name, "wb");

if (file) {

while ((bytes\_read = read(socket, buffer, BUFFER\_SIZE)) > 0) {

fwrite(buffer, 1, bytes\_read, file);

}

fclose(file);

printf("File %s downloaded.\n", file\_name);

} else {

perror("fopen");

}

}

int main(int argc, char \*argv[]) {

int client\_socket;

struct sockaddr\_un server\_addr;

if (argc < 3) {

fprintf(stderr, "Usage: %s <upload|download> <file\_name>\n", argv[0]);

exit(EXIT\_FAILURE);

}

// Create socket

client\_socket = socket(AF\_UNIX, SOCK\_STREAM, 0);

if (client\_socket == -1) {

perror("socket");

exit(EXIT\_FAILURE);

}

// Connect to server

memset(&server\_addr, 0, sizeof(struct sockaddr\_un));

server\_addr.sun\_family = AF\_UNIX;

strncpy(server\_addr.sun\_path, SOCKET\_PATH, sizeof(server\_addr.sun\_path) - 1);

if (connect(client\_socket, (struct sockaddr \*)&server\_addr, sizeof(struct sockaddr\_un)) == -1) {

perror("connect");

exit(EXIT\_FAILURE);

}

if (strcmp(argv[1], "upload") == 0) {

upload\_file(client\_socket, argv[2]);

} else if (strcmp(argv[1], "download") == 0) {

download\_file(client\_socket, argv[2]);

} else {

fprintf(stderr, "Unknown command: %s\n", argv[1]);

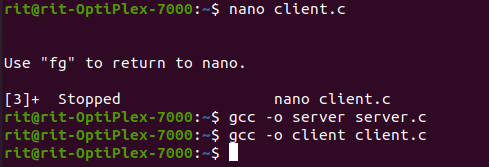
}

close(client\_socket);

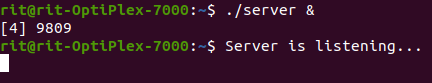
return 0;

}

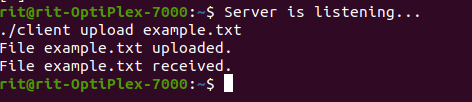
**Output:**



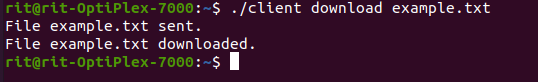
Running server in the background:



Uploding the file:



Downloading the file:



**Result:**

Thus the project done for creating a peer-to-peer file sharing application using IPC and program for this in C language. The above code for server and client was successfully executed.