**Taaha Hussain Khan**

**L1F21BSCS0917**

**Task 01:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

struct ThreadArgs {

    char \*name;

    int reg\_number;

};

void \*printInfo(void \*args) {

    struct ThreadArgs \*info = (struct ThreadArgs \*)args;

    printf("Name: %s\nRegistration Number: %d\n", info->name, info->reg\_number);

    pthread\_exit(NULL);

}

int main() {

    pthread\_t tid;

    struct ThreadArgs args;

    args.name = "Taaha Hussain Khan";

    args.reg\_number = 917;

    if (pthread\_create(&tid, NULL, printInfo, (void \*)&args) != 0) {

        fprintf(stderr, "Failed to create thread\n");

        return 1;

    }

    if (pthread\_join(tid, NULL) != 0) {

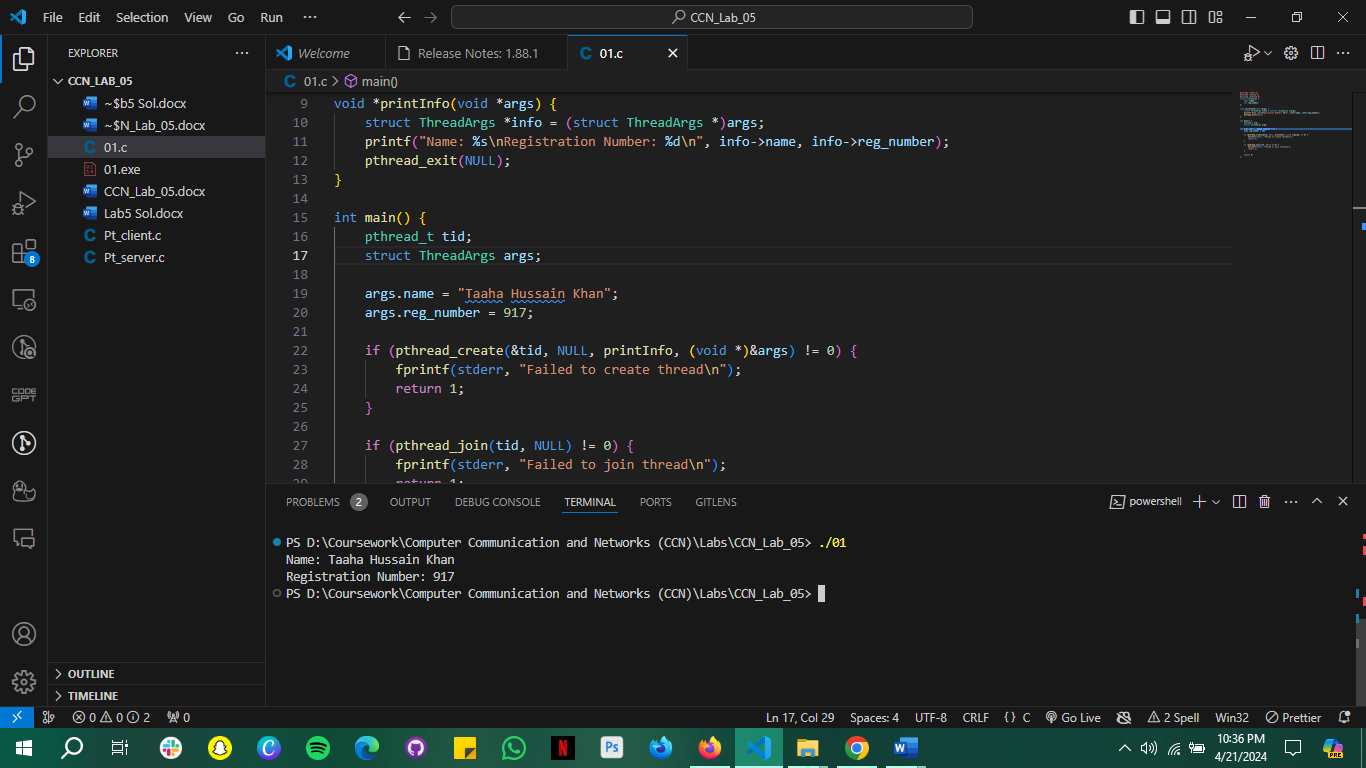
        fprintf(stderr, "Failed to join thread\n");

        return 1;

    }

    return 0;

}



**Task 2:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

struct ThreadArgs {

    char \*name;

    int reg\_number;

};

void \*printInfo(void \*args) {

    struct ThreadArgs \*info = (struct ThreadArgs \*)args;

    printf("Name: %s\nRegistration Number: %d\n", info->name, info->reg\_number);

    pthread\_exit(NULL);

}

int main() {

    pthread\_t tid;

    struct ThreadArgs args;

    args.name = "Taaha Hussain Khan";

    args.reg\_number = 917;

    if (pthread\_create(&tid, NULL, printInfo, (void \*)&args) != 0) {

        fprintf(stderr, "Failed to create thread\n");

        return 1;

    }

    if (pthread\_join(tid, NULL) != 0) {

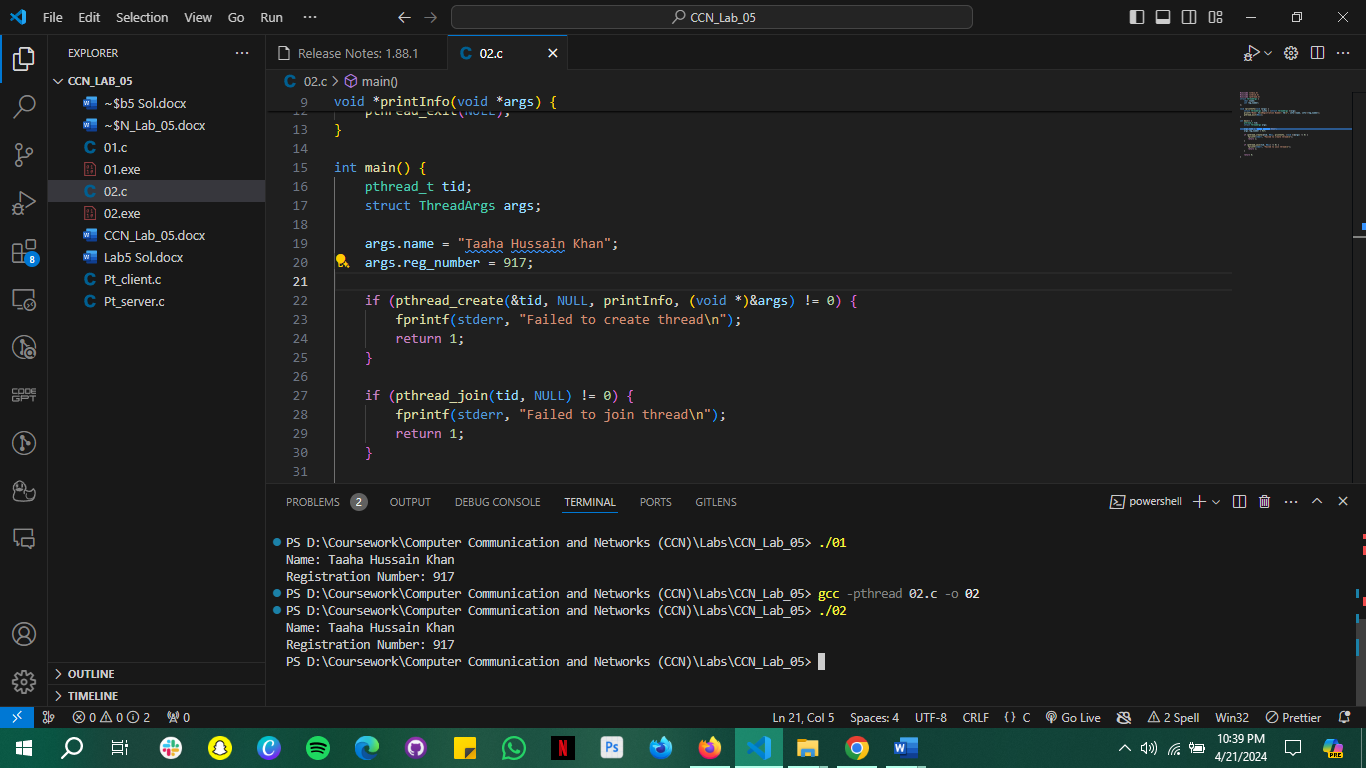
        fprintf(stderr, "Failed to join thread\n");

        return 1;

    }

    return 0;

}



**Task 3**

**Server:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <unistd.h>**

**#include <pthread.h>**

**#include <sys/socket.h>**

**#include <netinet/in.h>**

**#define PORT 8080**

**#define MAX\_CLIENTS 10**

**#define MAX\_FILES 3**

**char \*files[MAX\_FILES] = {"file1.txt", "file2.txt", "file3.txt"};**

**void \*handleClient(void \*arg) {**

**int client\_socket = \*((int \*)arg);**

**char buffer[1024] = {0};**

**struct sockaddr\_in addr;**

**socklen\_t addr\_size = sizeof(struct sockaddr\_in);**

**getpeername(client\_socket, (struct sockaddr \*)&addr, &addr\_size);**

**char ip\_address[INET\_ADDRSTRLEN];**

**inet\_ntop(AF\_INET, &addr.sin\_addr, ip\_address, INET\_ADDRSTRLEN);**

**int port = ntohs(addr.sin\_port);**

**sprintf(buffer, "IP Address: %s\nPort: %d\n", ip\_address, port);**

**send(client\_socket, buffer, strlen(buffer), 0);**

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

**sprintf(buffer, "Available files:\n");**

**for (int i = 0; i < MAX\_FILES; i++) {**

**strcat(buffer, files[i]);**

**strcat(buffer, "\n");**

**}**

**send(client\_socket, buffer, strlen(buffer), 0);**

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

**recv(client\_socket, buffer, 1024, 0);**

**FILE \*file = fopen(buffer, "r");**

**if (file == NULL) {**

**send(client\_socket, "File not found", strlen("File not found"), 0);**

**} else {**

**while (fgets(buffer, sizeof(buffer), file) != NULL) {**

**send(client\_socket, buffer, strlen(buffer), 0);**

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

**}**

**fclose(file);**

**}**

**send(client\_socket, "Do you want to download more files? (Yes/No): ", strlen("Do you want to download more files? (Yes/No): "), 0);**

**recv(client\_socket, buffer, 1024, 0);**

**if (strcmp(buffer, "Yes\n") == 0) {**

**handleClient(arg);**

**}**

**close(client\_socket);**

**pthread\_exit(NULL);**

**}**

**int main() {**

**int server\_socket, client\_socket;**

**struct sockaddr\_in server\_addr, client\_addr;**

**pthread\_t tid;**

**int opt = 1;**

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

**if (server\_socket == -1) {**

**perror("Socket creation failed");**

**exit(EXIT\_FAILURE);**

**}**

**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, MAX\_CLIENTS) < 0) {**

**perror("Listen failed");**

**exit(EXIT\_FAILURE);**

**}**

**while (1) {**

**socklen\_t addr\_size = sizeof(struct sockaddr\_in);**

**client\_socket = accept(server\_socket, (struct sockaddr \*)&client\_addr, &addr\_size);**

**if (client\_socket < 0) {**

**perror("Accept failed");**

**exit(EXIT\_FAILURE);**

**}**

**pthread\_create(&tid, NULL, handleClient, (void \*)&client\_socket);**

**}**

**close(server\_socket);**

**return 0;**

**}**

**Client:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <unistd.h>**

**#include <sys/socket.h>**

**#include <netinet/in.h>**

**#include <arpa/inet.h>**

**#define PORT 8080**

**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, "127.0.0.1", &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);**

**}**

**recv(client\_socket, buffer, 1024, 0);**

**printf("%s\n", buffer);**

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

**recv(client\_socket, buffer, 1024, 0);**

**printf("%s\n", buffer);**

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

**printf("Enter file name to download: ");**

**fgets(buffer, sizeof(buffer), stdin);**

**send(client\_socket, buffer, strlen(buffer), 0);**

**FILE \*file = fopen("downloaded\_file.txt", "w");**

**while (recv(client\_socket, buffer, 1024, 0) > 0) {**

**fputs(buffer, file);**

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

**}**

**fclose(file);**

**close(client\_socket);**

**return 0;**

**}**