Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та комп’ютерних технологій

Кафедра радіоелектронних і комп’ютерних систем

Звіт

про виконання лабораторних робіт №7

“Реалізація міжпроцесової взаємодії на основі інтерфейсу файлової системи”

Виконав

студент групи ФеІ-23

Гупало Мар’ян

Перевірив

ac. Сінькевич О. О.

Львів – 2019

Завдання №1.

Розробіть систему обміну даними про поточну температуру повітря для Linux і Windows XP з використанням відображуваної пам'яті

**Server-client.c**

#include <pthread.h>

#include <unistd.h>

#include <sys/mman.h>

#include <fcntl.h>

#include <stdio.h>

void\* server(void\* par)

{

int fdl;

int\* map;

fdl = open("tmp", O\_RDWR | O\_CREAT, 0644);

lseek(fdl, sizeof(int), SEEK\_SET);

write(fdl, "", 1);

map = (int\*)mmap(0, sizeof(int), PROT\_WRITE | PROT\_READ, MAP\_SHARED, fdl, 0);

close(fdl);

map[0] = 51;

while(1)

{

printf("Server temperature is : %i\n", map[0]);

if (map[0] == 0) break;

usleep(200);

map[0]--;

}

}

int index = 0;

pthread\_mutex\_t i = PTHREAD\_MUTEX\_INITIALIZER;

int getIndex(){

index++;

return index;

}

void\* client(void\* par) {

usleep(200);

int fdl;

int\* map;

pthread\_mutex\_lock(&i);

int index = getIndex();

pthread\_mutex\_unlock(&i);

fdl = open("./tmp", O\_RDONLY);

map = (int\*)mmap(0, sizeof(int), PROT\_READ, MAP\_SHARED, fdl, 0);

close(fdl);

while(1){

printf("client %i temperature is : %i\n", index, map[0]);

if (map[0] == 0)

break;

usleep(200);

}

}

int main() {

int clientCount = 3;

pthread\_t serverThread;

pthread\_t clientThreads[clientCount];

pthread\_create(&serverThread, NULL, server, NULL);

for (int i = 0; i < clientCount; ++i)

pthread\_create(&clientThreads[i], NULL, client, NULL);

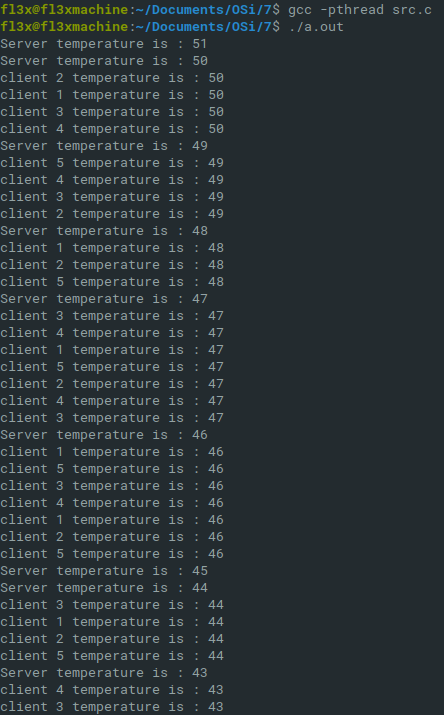
for (int i = 0; i < clientCount; ++i)

pthread\_join(clientThreads[i], NULL);

pthread\_join(serverThread, NULL);

return 0;

}



Завдання №2

Розробіть просту клієнт-серверну систему для Linux і Windows XP з використанням поіменованих каналів.

**Serv.c**

#include <unistd.h>

#include <stdio.h>

#include <sys/socket.h>

#include <stdlib.h>

#include <netinet/in.h>

#include <string.h>

#define PORT 8080

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

{

int server\_fd, new\_socket, valread;

struct sockaddr\_in address;

int opt = 1;

int addrlen = sizeof(address);

char buffer[1024] = {0};

char \*hello = "Hello from server";

// Creating socket file descriptor

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0)

{

perror("socket failed");

exit(EXIT\_FAILURE);

}

// Forcefully attaching socket to the port 8080

if (setsockopt(server\_fd, SOL\_SOCKET, SO\_REUSEADDR | SO\_REUSEPORT,

&opt, sizeof(opt)))

{

perror("setsockopt");

exit(EXIT\_FAILURE);

}

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons( PORT );

// Forcefully attaching socket to the port 8080

if (bind(server\_fd, (struct sockaddr \*)&address,

sizeof(address))<0)

{

perror("bind failed");

exit(EXIT\_FAILURE);

}

if (listen(server\_fd, 3) < 0)

{

perror("listen");

exit(EXIT\_FAILURE);

}

if ((new\_socket = accept(server\_fd, (struct sockaddr \*)&address,

(socklen\_t\*)&addrlen))<0)

{

perror("accept");

exit(EXIT\_FAILURE);

}

valread = read( new\_socket , buffer, 1024);

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

send(new\_socket , hello , strlen(hello) , 0 );

printf("Hello message sent\n");

return 0;

}

**Client.c**

#include <stdio.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <string.h>

#define PORT 8080

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

{

    int sock = 0, valread;

    struct sockaddr\_in serv\_addr;

    char \*hello = "Hello from client";

    char buffer[1024] = {0};

    if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

    {

        printf("\n Socket creation error \n");

        return -1;

    }

    serv\_addr.sin\_family = AF\_INET;

    serv\_addr.sin\_port = htons(PORT);

    // Convert IPv4 and IPv6 addresses from text to binary form

    if(inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr)<=0)

    {

        printf("\nInvalid address/ Address not supported \n");

        return -1;

    }

    if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0)

    {

        printf("\nConnection Failed \n");

        return -1;

    }

    send(sock , hello , strlen(hello) , 0 );

    printf("Hello message sent\n");

    valread = read( sock , buffer, 1024);

    printf("%s\n",buffer );

    return 0;

}



Висновок: на даній лабораторній роботі я дізнався про роботу з потоками, портами та реалізував сервер-клієнтську програму.