**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

по дисциплине «Программирование сетевых приложений»

на тему: «**Организация распределённых вычислений с**

**использованием сокет *TCP*/*IP* средствами *LinuxAPI***»

Выполнил: студент гр. ИТИ-41

Гончар А.С.

Принял: ассистент

Карабчикова Е.А.

Гомель 2020

**Цель работы**: изучение сетевого взаимодействия с использованием сокет *TCP*/*IP* средствами *LinuxAPI*.

Задание для 5 варианта: Распределенная сортировка, для сбора от клиентов использовать mergesort, на клиентах использовать сортировку вставкой.

**Результат выполнения:**

В ходе выполнения лабораторной работы были разработаны клиент и сервер:

– сервер для считывания данных с файла, разделения на порции, передачи на остальные клиенты и сборки результирующих частей;

– клиент, представляющий собой юнит для вычисления порции данных: сортировки последовательности, переданной с основного клиента.

Для анализа эффективности расчетов на нескольких узлах, были произведены вычисления и созданы графики, описывающие зависимость скорости вычисления от количества узлов, а также от количества исходных данных.

На графике 1 представлены графики зависимости времени выполнения сортировки от количества данных, при расчетах на разном количестве узлов.

График 1 – График зависимости количества данных от количества узлов

**Вывод**: в ходе выполнения лабораторной работы разработано серверное приложение нахождения площади фигуры с использованием сокетов *TCP*/*IP* в среде *LinuxAPI*.

**ПРИЛОЖЕНИЕ А**

Листинг программы

Файл ***ServerApp.cpp***:

#include <stdio.h>

#include <stdlib.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <netdb.h>

#include <unistd.h>

#include <string.h>

#include <arpa/inet.h>

#include <stdbool.h>

#include <vector>

#include <ctime>

#include <sys/select.h>

#include <iostream>

using namespace std;

vector<int> merge(vector<int> a1, vector<int> a2) {

const int a1l = a1.size();

const int a2l = a2.size();

const int a3l = a1l + a2l;

vector<int> a3(a3l);

int i = 0, j = 0;

for (int k = 0; k < a3l; k++) {

if (i > a1l - 1) {

int a = a2[j];

a3[k] = a;

j++;

}

else if (j > a2l - 1) {

int a = a1[i];

a3[k] = a;

i++;

}

else if (a1[i] < a2[j]) {

int a = a1[i];

a3[k] = a;

i++;

}

else {

int b = a2[j];

a3[k] = b;

j++;

}

}

return a3;

}

int nclients = 0;

vector<vector<int>> arr;

int CountOfClients;

int ArraySize;

int PartSize;

long start\_time, end\_time;

void final() {

vector<int> result(0);

for (int i = 0; i < CountOfClients; i++) {

result = merge(result, arr[i]);

}

end\_time = clock() - start\_time;

cout << end\_time;

cout << " ms\n";

for (vector<int>::iterator it = result.begin(); it != result.end(); ++it) {

cout << \*it;

cout << "\n";

}

}

int main() {

// create a socket

int server\_socket = 0;

int checkSelect, checkBind, checkListen, checkSend, checkSockopt;

fd\_set master; // create a master set

fd\_set clients; // create set for clients

printf("TCP SERVER \n");

cout << "Enter the number of clients:\n";

cin >> CountOfClients;

do {

cout << "Enter the number of items:\n";

cin >> ArraySize;

} while ((ArraySize % CountOfClients) != 0);

PartSize = ArraySize / CountOfClients;

char buff[1024];

setlocale(0, "");

arr = vector<vector<int>>(CountOfClients, vector<int>(PartSize));

for (int i = 0; i < CountOfClients; ++i) {

for (int j = 0; j < PartSize; ++j) {

arr[i][j] = rand() % 1000 + 1;

}

}

// define the server address

struct sockaddr\_in server\_address;

struct sockaddr\_in client\_address;

FD\_ZERO(&master); // zero the set

// (Domain, Type of socket, Protocol)

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

if(server\_socket == -1){

perror("socket");

exit(1);

}

// bind the socket to our specified IP and port

server\_address.sin\_family = AF\_INET; // // knows what type of server\_address

server\_address.sin\_port = htons(9002); // htons() converts data format so that our struct can understand the port number

server\_address.sin\_addr.s\_addr = INADDR\_ANY; // bind to any available address

memset(server\_address.sin\_zero, '\0', sizeof server\_address.sin\_zero); // for padding, to make it the same size as struct sockaddr

int yes = 1;

checkSockopt = setsockopt(server\_socket, SOL\_SOCKET, SO\_REUSEADDR, &yes, sizeof(int));

if(checkSockopt == -1){

perror("setsockopt");

exit(1);

}

checkBind = bind(server\_socket, (struct sockaddr \*) &server\_address, sizeof(struct sockaddr));

if(checkBind == -1){

perror("unable to bind");

exit(1);

}

// begin listening for clients attempting to connect

checkListen = listen(server\_socket, CountOfClients);

if(checkListen == -1){

perror("listen");

exit(1);

}

fflush(stdout);

//printf("server socket: \t%d\n", server\_socket);

FD\_SET(server\_socket, &master); // sets the bit for the server\_socket in the master socket set

printf("Ожидание подключений...\n");

//My

socklen\_t addrlen;

int client\_socket;

addrlen = sizeof(struct sockaddr\_in);

vector<int> client\_sockets(CountOfClients);

while(client\_socket = accept(server\_socket, (struct sockaddr \*)&client\_address, &addrlen)){

client\_sockets[nclients] = client\_socket;

nclients++; // увеличиваем счетчик подключившихся клиентов

// пытаемся получить имя хоста

hostent\* hst;

hst = gethostbyaddr((char\*)&client\_address.sin\_addr.s\_addr, 4, AF\_INET);

// вывод сведений о клиенте

printf("+%s [%s] new connect!\n",

(hst) ? hst->h\_name : "", inet\_ntoa(client\_address.sin\_addr));

if (nclients == CountOfClients) {

start\_time = clock();

for (int i = 0; i < CountOfClients; i++) {

send(client\_sockets[i], (char\*)&PartSize, sizeof(int), 0);

send(client\_sockets[i], (char\*)&arr[i][0], sizeof(int) \* PartSize, 0);

int bytes\_recv;

bytes\_recv = recv(client\_sockets[i], (char\*)&arr[i][0], sizeof(int) \* PartSize, 0);

nclients--; // уменьшаем счетчик активных клиентов

printf("-disconnect\n");

{

// закрываем сокет

close(client\_sockets[i]);

}

}

final();

}

}

//EndMy

close(server\_socket);

return 0;

}

Файл ***ClientApp.cpp***:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <unistd.h>

#include <string.h>

#include <arpa/inet.h>

#include <vector>

#include <stdbool.h>

#include <iostream>

using namespace std;

char SERVERADDR[11];

int main() {

// create a socket

int network\_socket = 0;

fd\_set master; // create a master set

fd\_set clients; // create set for clients

printf("TCP CLIENT\n");

int choose;

cout << "Choose:\n1.Local IP\n2.Enter IP\n";

cin >> choose;

switch (choose)

{

case 1:

strcpy(SERVERADDR, "127.0.0.1");

break;

case 2:

cout << "Enter server's ip address:";

cin >> SERVERADDR;

break;

}

// define the server address

struct sockaddr\_in server\_address;

network\_socket = socket(AF\_INET, SOCK\_STREAM, 0); // (Domain, Type of socket, Protocol)

if(network\_socket == -1){

perror("socket");

exit(1);

}

//

server\_address.sin\_family = AF\_INET; // knows what type of server\_address

server\_address.sin\_port = htons(9002); // htons() converts data format so that our struct can understand the port number

server\_address.sin\_addr.s\_addr = inet\_addr(SERVERADDR); // bind to any available address

memset(server\_address.sin\_zero, '\0', sizeof(server\_address.sin\_zero)); // for padding, to make it the same size as struct sockaddr

// connect with server socket

int connection\_status = connect(network\_socket, (struct sockaddr\*) &server\_address, sizeof(server\_address));

// check for error with connection

if(connection\_status == -1){

perror("connect");

exit(-1);

}else{

printf("Connected to remote socket \n");

}

FD\_ZERO(&master); // zero the sets

FD\_ZERO(&clients);

FD\_SET(0, &master); // add zero to the set so that we can watch for input

FD\_SET(network\_socket, &master); // add the connected socket

//Получение размера массива

int arraysize = 0;

int nsize;

nsize = recv(network\_socket, (char\*)&arraysize, sizeof(int), 0);

//Получение массива

vector<int> res(arraysize);

nsize = recv(network\_socket, (char\*)&res[0], arraysize \* sizeof(int), 0);

//Сортировка

int tmp;

for (int i = 1; i < arraysize; i++)

for (int j = i; j > 0 && res[j - 1] > res[j]; j--)

{

tmp = res[j - 1];

res[j - 1] = res[j];

res[j] = tmp;

}

//Отправка отсортированного массива обратно на сервер

send(network\_socket, (char\*)&res[0], sizeof(int) \* arraysize, 0);

/\*for (const auto& e : res) {

cout << e << endl;

}\*/

// Need to print a welcome message to the client when they first connect to the server

// Give clients the ability to choose a username upon connecting

close(network\_socket);

return 0;

}