МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационные системы

Бариев Эмин Юсуфович

Институт информационных технологий и управления в технических системах

курс 4 группа ИС/б-16-2

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №4

По дисциплине: «ТРСиПВ»

По теме: «ИССЛЕДОВАНИЕ ВЗАИМОДЕЙСТВИЙ РАСПРЕДЕЛЕННЫХ ПРОЦЕССОВ ТИПА «КЛИЕНТ-СЕРВЕР»»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

ст.преподаватель Дрозин А. Ю.

(должность) (подпись) (инициалы,фамилия)

Севастополь 2019

1 ЦЕЛЬ РАБОТЫ

Исследовать механизм взаимодействия распределено выполняющихся параллельных процессов типа «клиент-сервер».

2 ПОСТАНОВКА ЗАДАЧИ

Вариант – 1

В состав вычислительного кластера входит три хоста, один из которых реализует функции сервера, два остальных – клиентов.

Сервер разграничивает доступ к трем общим ресурсам – нерешенным, хранящим общую вырученную сумму от продажи товаров, общее количество товаров и остальных товаров. Доступ к ресурсам осуществляется в произвольном порядке, все ресурсы разделяются между клиентами по отдельности. Реализована процедура выделяющая ресурсы (путем передачи сообщения) в использование клиентам. Реализовать серверный процесс, который разграничивает доступ клиентов к этой процедуре (процедурам) и к ресурсам. Реализацию сервера выполнять в соответствии со схемой управления, использующую рассылку сообщений.

3 ХОД РАБОТЫ

3.1 Код программы на языке С++

#include <iostream>

#include <ctime>

#include <mpi.h>

**using** **namespace** std;

**enum** {REQUEST = 1, RELEASE = 2, CONNECT = 3, CONFIRM = 4, DISCONNECT = 5, BUSY = 6, CLIENTS = 7, FREE = 8};

**const** **int** SERVER = 0;

**const** **int** RESOURCE = 1;

**int** procNum;

**int** procRank;

**int** clients = 0;

MPI\_Status status;

MPI\_Comm comm;

**void** server();

**void** client();

**void** resource();

**void** getResurce(**int** resource);

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

MPI\_Init(&argc, &argv);

comm = MPI\_COMM\_WORLD;

MPI\_Comm\_size(comm, &procNum);

MPI\_Comm\_rank(comm, &procRank);

**if**(procRank == SERVER) server();

**else** **if**(procRank == RESOURCE) resource();

**else** client();

MPI\_Barrier(comm);

MPI\_Finalize();

**return** 0;

}

**void** resource(){

**int** resources[] = {10, 2, 5};

**int** resourcesInUse[] = {0, 0, 0};

**int** infoBuf;

**int** numberResource;

**int** bufForResource;

**int** fail\_buffer = -1;

**do**{

MPI\_Recv(&infoBuf, 1, MPI\_INT, SERVER, MPI\_ANY\_TAG, comm, &status);

**switch**(status.MPI\_TAG){

**case** CLIENTS:

clients = infoBuf;

**break**;

**case** REQUEST :

numberResource = infoBuf;

cout<<"Запрос сервера на ресурс "<<numberResource<<";"<<endl;;

**if**(resourcesInUse[numberResource] != 0){

MPI\_Send(&fail\_buffer, 1, MPI\_INT, SERVER, BUSY, comm);

**break**;

}

resourcesInUse[numberResource] = 1;

MPI\_Send(&resources[numberResource], 1, MPI\_INT, SERVER, CONFIRM, comm);

**break**;

**case** RELEASE :

numberResource = infoBuf;

cout<<"Запрос сервера на особождение ресурса "<<numberResource<<";";

resourcesInUse[numberResource] = 0;

cout<<"Ресурс "<<numberResource<<" освобожден;"<<endl;

MPI\_Send(&fail\_buffer, 1, MPI\_INT, SERVER, FREE, comm);

**break**;

}

}**while**(clients);

}

**void** server(){

**int** numberResource;

**int** queueProc=-1, queueResource=-1;

**int** bufForResource;

**int** clientSource;

**do**{

MPI\_Recv(&numberResource, 1, MPI\_INT, MPI\_ANY\_SOURCE, MPI\_ANY\_TAG, comm, &status);

clientSource = status.MPI\_SOURCE;

**switch**(status.MPI\_TAG){

**case** CONNECT :

cout<<"Клиент"<<clientSource - 1<<" в системе;"<<endl;

clients++;

MPI\_Send(&clients, 1, MPI\_INT, RESOURCE, CLIENTS, comm);

**break**;

**case** DISCONNECT :

cout<<"Клиент"<<clientSource - 1<<" вышел из системы;"<<endl;

clients--;

MPI\_Send(&clients, 1, MPI\_INT, RESOURCE, CLIENTS, comm);

**break**;

**case** REQUEST :

cout<<"Клиент"<<clientSource - 1<<" запрашивает ресурс "<<numberResource<<";"<<endl;

MPI\_Send(&numberResource, 1, MPI\_INT, RESOURCE, REQUEST, comm);

MPI\_Recv(&bufForResource, 1, MPI\_INT, RESOURCE, MPI\_ANY\_TAG, comm, &status);

**if**(status.MPI\_TAG == BUSY){

queueProc = clientSource;

queueResource = numberResource;

cout<<"Для Клиента"<<clientSource - 1<<" ресурс "<<numberResource<<" занят;"<<endl;

**break**;

}

MPI\_Send(&bufForResource, 1, MPI\_INT, clientSource, CONFIRM, comm);

**break**;

**case** RELEASE :

cout<<"Клиент"<<clientSource-1<<" освободил ресурс "<<numberResource<<";"<<endl;

MPI\_Send(&numberResource, 1, MPI\_INT, RESOURCE, RELEASE, comm);

MPI\_Recv(&bufForResource, 1, MPI\_INT, RESOURCE, FREE, comm, MPI\_STATUS\_IGNORE);

**if**(queueResource == numberResource){

MPI\_Send(&numberResource, 1, MPI\_INT, RESOURCE, REQUEST, comm);

MPI\_Recv(&bufForResource, 1, MPI\_INT, RESOURCE, MPI\_ANY\_TAG, comm, &status);

MPI\_Send(&bufForResource, 1, MPI\_INT, queueProc, CONFIRM, comm);

queueProc=-1;

queueResource=-1;

**break**;

}

**break**;

}

}**while**(clients);

}

**void** getResurce(**int** resource){

**int** bufForResource;

MPI\_Send(&resource, 1, MPI\_INT, SERVER, REQUEST, comm);

MPI\_Recv(&bufForResource, 1, MPI\_INT, SERVER, CONFIRM, comm, &status);

cout<<"Клиент"<<procRank-1<<"получил ресурс "<<bufForResource<<";"<<endl;

MPI\_Send(&resource, 1, MPI\_INT, SERVER, RELEASE, comm);

}

**void** client(){

**int** numberResurce;

srand(10);

**if** (procRank == 1) srand(10);

**else** srand(11);

numberResurce = rand() % 3;

MPI\_Send(&numberResurce, 1, MPI\_INT, SERVER, CONNECT, comm);

**for**(**int** i = 0; i < 4; i++){

numberResurce = rand() % 3;

getResurce(numberResurce);

}

MPI\_Send(&numberResurce, 1, MPI\_INT, SERVER, DISCONNECT, comm);

}

3.2 Результаты выполнения программы

**Клиент1 в системе;**

**Клиент1 запрашивает ресурс 1;**

**Запрос сервера на ресурс 1;**

**Клиент2 в системе;**

**Клиент2 запрашивает ресурс 1;**

**Для Клиента2 ресурс 1 занят;**

**Запрос сервера на ресурс 1;**

**Клиент1получил ресурс 2;**

**Клиент1 освободил ресурс 1;**

**Клиент1 запрашивает ресурс 2;**

**Запрос сервера на особождение ресурса 1;Ресурс 1 освобожден;**

**Запрос сервера на ресурс 1;**

**Запрос сервера на ресурс 2;**

**Клиент1получил ресурс 5;**

**Клиент2Клиент1 освободил ресурс 2;**

**получил ресурс 2;**

**Запрос сервера на особождение ресурса 2;Ресурс 2 освобожден;**

**Клиент2 освободил ресурс 1;**

**Запрос сервера на особождение ресурса 1;Ресурс 1 освобожден;**

**Клиент1 запрашивает ресурс 2;**

**КлиентЗапрос сервера на ресурс 2;**

**Запрос сервера на ресурс 2;**

**Запрос сервера на особождение ресурса 2;Ресурс Клиент1 получил ресурс 5;**

**2 запрашивает ресурс 2;**

**Для Клиента2 ресурс 2 занят;**

**Клиент1 освободил ресурс 2;**

**2 освобожден;**

**Запрос сервера на ресурс 2;**

**Клиент2получил ресурс 5;**

**Клиент1 запрашивает ресурс 0;**

**Клиент2 освободил ресурс 2;**

**Запрос сервера на ресурс 0;**

**Запрос сервера на особождение ресурса Клиент1получил ресурс 10;**

**2;Ресурс 2 освобожден;**

**Клиент1 освободил ресурс 0;**

**Запрос сервера на особождение ресурса 0;Ресурс 0 освобожден;**

**Запрос сервера на ресурс 2;**

**Клиент2получил ресурс 5;**

**Клиент2 запрашивает ресурс 2;**

**Клиент1 вышел из системы;**

**Клиент2 освободил ресурс 2;**

**Запрос сервера на особождение ресурса 2;Ресурс 2 освобожден;**

**Клиент2 запрашивает ресурс 0;**

**Клиент2получил ресурс 10;**

**Запрос сервера на ресурс 0;**

**Клиент2 освободил ресурс 0;**

**КлиентЗапрос сервера на особождение ресурса 0;Ресурс 0 освобожден;**

**2 вышел из системы;**

**Program ended with exit code: 0**

ВЫВОДЫ

В ходе выполнения лабораторной работы был исследован механизм взаимодействия распределено выполняющихся параллельных процессов типа «клиент-сервер».

Была создана программа, реализующая взаимодействия распределено выполняющихся параллельных процессов типа «клиент-сервер».