|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 **Информатика и вычислительная техника**

**ОТЧЕТ**

**по студенческому практикуму**

**Дисциплина:** Организация ЭВМ и систем

**Название:** Разработка и отладка программ в вычислительном комплексе Тераграф

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Попов. А. Ю

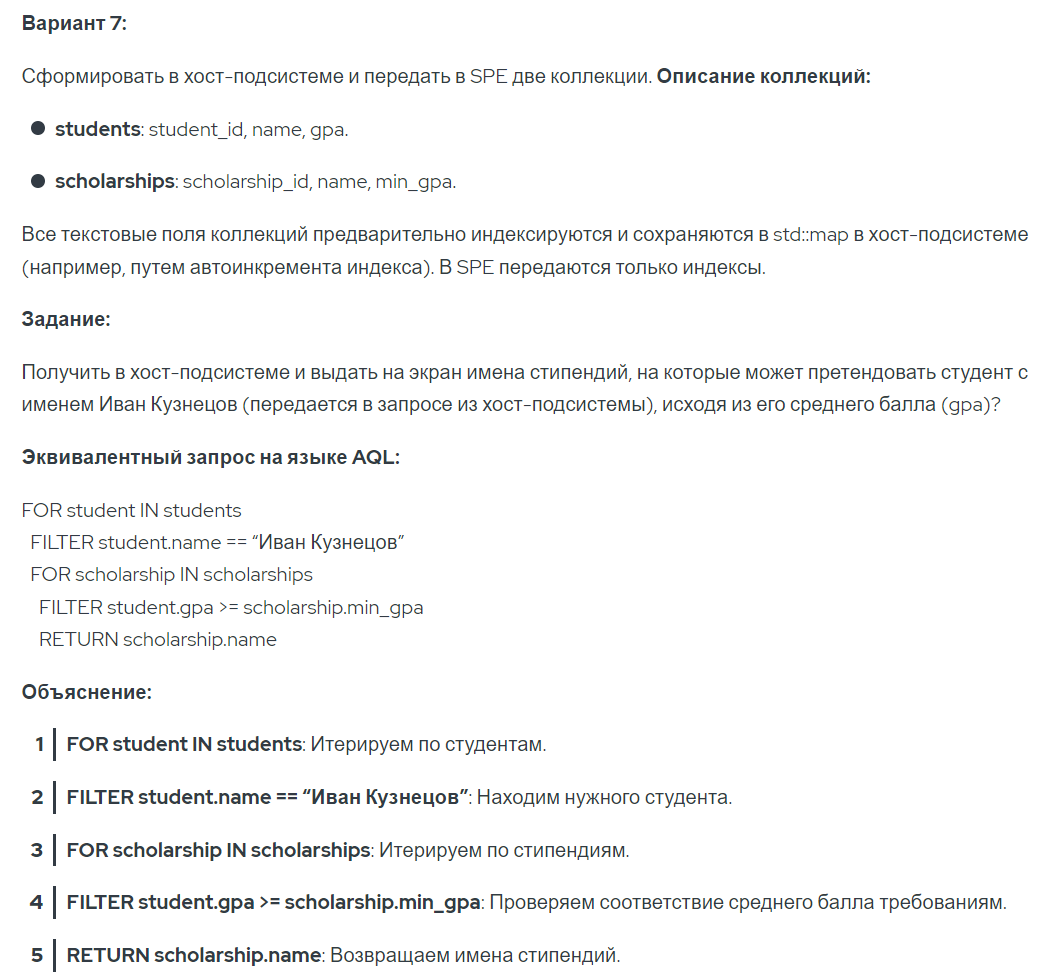
(Подпись, дата) (И.О. Фамилия)

Студент гр. ИУ7-55Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Талышева О.Н.

(Подпись, дата) (И.О. Фамилия)

Москва, 2024

**Задание**



**Код**

/host/src/host\_main.cpp

#**include** <iostream>

#**include** <iterator>

#**include** <string>

#**include** <regex>

#**include** <sstream>

#**include** <fstream>

#**include** <ctime>

#**include** <vector>

#**include** "host\_main.h"

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

string target\_name="Иван Кузнецов";

vector<string> student\_names = {

"Алексей Иванов",

"Мария Смирнова",

"Дмитрий Кузнецов",

"Екатерина Соколова",

"Иван Кузнецов", //\*

"Иван Попов",

"Ольга Лебедева",

"Николай Новиков",

"София Морозова",

"Павел Васильев"

};

vector<uint32\_t> min\_gpa = {

1,

2,

3,

4,

5,

6,

7,

8,

9,

1

};

vector<string> scholarship\_names = {

"scholarship\_1",

"scholarship\_2",

"scholarship\_3",

"scholarship\_4",

"scholarship\_5",

"scholarship\_6",

"scholarship\_7",

"scholarship\_8",

"scholarship\_9",

"scholarship\_10",

};

map<string, uint32\_t> student\_name\_to\_index;

map<uint32\_t, string> index\_to\_student\_name;

map<string, uint32\_t> scholarship\_name\_to\_index;

map<uint32\_t, string> index\_to\_scholarship\_name;

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

{

ofstream **log**("lab2.log");

gpc \*gpc64\_inst;

//Инициализация gpc

**if** (argc<3) {

log<<"Использование: host\_main <путь к файлу rawbinary> <gpc>"<<endl;

**return** -1;

}

//Захват ядра gpc и запись sw\_kernel

gpc64\_inst = **new** gpc(argv[2]);

log<<"Открывается доступ к "<<gpc64\_inst->gpc\_dev\_path<<endl;

**if** (gpc64\_inst->load\_swk(argv[1])==0) {

log<<"Программное ядро загружено из файла "<<argv[1]<<endl;

}

**else** {

log<<"Ошибка загрузки sw\_kernel файла << argv[1]"<<endl;

**return** -1;

}

// init

**for** (uint32\_t i=0; i<student\_names.size(); i++) {

student\_name\_to\_index[student\_names[i]] = i;

index\_to\_student\_name[i] = student\_names[i];

}

**for** (uint32\_t i=0; i<scholarship\_names.size(); i++) {

scholarship\_name\_to\_index[scholarship\_names[i]] = i;

index\_to\_scholarship\_name[i] = scholarship\_names[i];

}

// update

gpc64\_inst->start(\_\_scholarship\_\_(update\_student)); //обработчик вставки

**for** (uint32\_t i=0; i<student\_names.size(); i++) {

gpc64\_inst->mq\_send(students::key{.student\_id=i, .name\_hashed=i});

gpc64\_inst->mq\_send(students::val{.gpa=min\_gpa[i]});

}

gpc64\_inst->mq\_send(-1ull);

gpc64\_inst->start(\_\_scholarship\_\_(update\_scholarship)); //обработчик вставки

**for** (uint32\_t i=0; i<scholarship\_names.size(); i++) {

gpc64\_inst->mq\_send(scholarships::key{.scholarship\_id=i, .min\_gpa=min\_gpa[i]});

gpc64\_inst->mq\_send(scholarships::val{.name\_hashed=i});

}

gpc64\_inst->mq\_send(-1ull);

// select

gpc64\_inst->start(\_\_scholarship\_\_(select)); //обработчик запроса поиска

gpc64\_inst->mq\_send(student\_name\_to\_index[target\_name]);

uint64\_t result\_student = gpc64\_inst->mq\_receive();

uint32\_t target\_gpa = students::val::from\_int(result\_student).gpa;

// gpc64\_inst->mq\_send(5); // Test other student

// gpc64\_inst->mq\_send(student\_names.size()+1); // Test no student

cout << "Scholarship for " << target\_name << " with gpa=" << target\_gpa << "\n";

**while** (1) {

uint64\_t result = gpc64\_inst->mq\_receive();

**if** (result!=-1ull) {

cout << index\_to\_scholarship\_name[scholarships::val::from\_int(result).name\_hashed] << "\n";

} **else** {

**break**;

}

}

log << "Выход!" << endl;

**return** 0;

}

/include/common\_struct.h

#ifndef COMMON\_STRUCT

#define COMMON\_STRUCT

#ifdef \_\_riscv64\_\_

#include "map.h"

#endif

#include "compose\_keys.hxx"

// Номера структур данных в SPE

enum Structures : uint32\_t {

null = 0, // Нулевая структура не используется

students\_pnum = 1, // Таблица 1

scholarships\_pnum = 2 // Таблица 2

};

#ifdef \_\_riscv64\_\_

// Задание даипазонов и курсоров

template <typename Range>

struct reverse {

Range r;

[[gnu::always\_inline]] reverse(Range r) : r(r) {}

[[gnu::always\_inline]] auto begin() { return r.rbegin(); }

[[gnu::always\_inline]] auto end() { return r.rend(); }

};

template <typename K, typename V>

struct Handle {

bool ret\_val;

K k{get\_result\_key<K>()};

V v{get\_result\_value<V>()};

[[gnu::always\_inline]] Handle(bool ret\_val) : ret\_val(ret\_val) {}

[[gnu::always\_inline]] operator bool() const { return ret\_val; }

[[gnu::always\_inline]] K key() const { return k; }

[[gnu::always\_inline]] V value() const { return v; }

};

#endif

//////////////////////////////////////

// Описание формата ключа и значения

//////////////////////////////////////

struct students {

using vertex\_t = uint32\_t;

int struct\_number;

constexpr students(int struct\_number) : struct\_number(struct\_number) {}

static const uint32\_t idx\_bits = 32;

static const uint32\_t idx\_max = (1ull << idx\_bits) - 1;

static const uint32\_t idx\_min = idx\_max;

STRUCT(key) {

uint32\_t student\_id : idx\_bits;

uint32\_t name\_hashed : 32;

};

STRUCT(val) {

uint32\_t gpa : 32;

uint32\_t not\_used : 32;

};

// Обязательная типизация

#ifdef \_\_riscv64\_\_

DEFINE\_DEFAULT\_KEYVAL(key, val)

#endif

};

constexpr students STUDENTS(Structures::students\_pnum);

struct scholarships {

using vertex\_t = uint32\_t;

int struct\_number;

constexpr scholarships(int struct\_number) : struct\_number(struct\_number) {}

static const uint32\_t idx\_bits = 32;

static const uint32\_t idx\_max = (1ull << idx\_bits) - 1;

static const uint32\_t idx\_min = idx\_max;

STRUCT(key) {

uint32\_t scholarship\_id : idx\_bits;

uint32\_t min\_gpa : 32;

};

STRUCT(val) {

uint32\_t name\_hashed : 32;

uint32\_t not\_used : 32;

};

// Обязательная типизация

#ifdef \_\_riscv64\_\_

DEFINE\_DEFAULT\_KEYVAL(key, val)

#endif

};

constexpr scholarships SCHOLARSHIPS(Structures::scholarships\_pnum);

#endif // COMMON\_STRUCT

/sw-kernel/src/sw\_kernel\_main.cpp

#include <stdlib.h>

#include <ctime>

#include "lnh64.hxx"

#include "gpc\_io\_swk.h"

#include "gpc\_handlers.h"

#include "common\_struct.h"

#include "compose\_keys.hxx"

#define \_\_fast\_recall\_\_

extern lnh lnh\_core;

volatile unsigned int scholarship\_source;

int main(void) {

/////////////////////////////////////////////////////////

// Main Event Loop

/////////////////////////////////////////////////////////

//Leonhard driver structure should be initialised

lnh\_init();

for (;;) {

//Wait for scholarship

scholarship\_source = wait\_scholarship();

switch(scholarship\_source) {

/////////////////////////////////////////////

// Measure GPN operation frequency

/////////////////////////////////////////////

case \_\_scholarship\_\_(update\_student) : update\_student(); break;

case \_\_scholarship\_\_(update\_scholarship) : update\_scholarship(); break;

case \_\_scholarship\_\_(select) : select(); break;

}

set\_gpc\_state(READY);

}

}

//-------------------------------------------------------------

// Вставка ключа и значения в структуру

//-------------------------------------------------------------

void update\_student() {

while(1){

students::key key=students::key::from\_int(mq\_receive());

if (key==-1ull) break;

students::val val=students::val::from\_int(mq\_receive());

STUDENTS.ins\_async(key,val); //Вставка в таблицу с типизацией uint64\_t

}

}

void update\_scholarship() {

while(1){

scholarships::key key=scholarships::key::from\_int(mq\_receive());

if (key==-1ull) break;

scholarships::val val=scholarships::val::from\_int(mq\_receive());

SCHOLARSHIPS.ins\_async(key,val); //Вставка в таблицу с типизацией uint64\_t

}

}

//-------------------------------------------------------------

// Передать все роли пользователя и время доступа

//-------------------------------------------------------------

void select() {

while(1){

uint32\_t qname = mq\_receive();

if (qname==-1) break;

auto student = STUDENTS.nsm(students::key{.student\_id=students::idx\_min, .name\_hashed=qname});

if (student && student.key().name\_hashed==qname)

{

uint32\_t target\_gpa = student.value().gpa;

mq\_send(student.value());

auto scholarship = SCHOLARSHIPS.nsm(scholarships::key{.scholarship\_id=scholarships::idx\_min, .min\_gpa=target\_gpa});

while (scholarship) {

mq\_send(scholarship.value());

scholarship = SCHOLARSHIPS.nsm(scholarship.key());

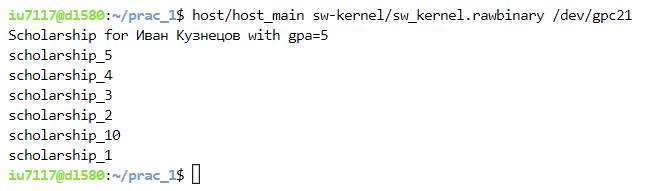
}

}

mq\_send(-1ull);

}

}**Пример работы программы**



**Выводы**

В ходе практикума была изучена типовая структура двух взаимодействующих программ: хост-подсистемы и программного ядра sw\_kernel. Также была разработана программа для хост-подсистемы и обработчик для программного ядра, который выполняет задачи, указанные в индивидуальном задании.