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

Вариант 7:

Сформировать в хост-подсистеме и передать в SPE две коллекции. **Описание коллекций:**

- **students**: student_id, name, gpa.
- **scholarships**: scholarship_id, name, min_gpa.

Все текстовые поля коллекций предварительно индексируются и сохраняются в std::map в хост-подсистеме (например, путем автоинкремента индекса). В SPE передаются только индексы.

Задание:

Получить в хост-подсистеме и выдать на экран имена стипендий, на которые может претендовать студент с именем Иван Кузнецов (передается в запросе из хост-подсистемы), исходя из его среднего балла (gpa)?

Эквивалентный запрос на языке AQL:

```
FOR student IN students
 FILTER student.name == "Иван Кузнецов"
 FOR scholarship IN scholarships
  FILTER student.gpa >= scholarship.min_gpa
  RETURN scholarship.name
```

Объяснение:

1. **FOR student IN students**: Итерируем по студентам.
2. **FILTER student.name == "Иван Кузнецов"**: Находим нужного студента.
3. **FOR scholarship IN scholarships**: Итерируем по стипендиям.
4. **FILTER student.gpa >= scholarship.min_gpa**: Проверяем соответствие среднего балла требованиям.
5. **RETURN scholarship.name**: Возвращаем имена стипендий.

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

```
iu7117@dl580:~/prac_1$ host/host_main sw-kernel/sw_kernel.rawbinary /dev/gpc21
Scholarship for Иван Кузнецов with gpa=5
scholarship_5
scholarship_4
scholarship_3
scholarship_2
scholarship_10
scholarship_1
iu7117@dl580:~/prac_1$
```

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.