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» (ИУ)
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

Отчет по лабораторной работе №3 по курсу «Архитектура электронно-вычислительных машин»

Разработка и отладка программ в вычислительном комплексе Тераграф

Группа: ИУ7-53Б

Студент: _____ Дьяченко А. А.
(Подпись, дата) (Фамилия И. О.)

Преподаватель: _____ Ибрагимов С. В.
(Подпись, дата) (Фамилия И. О.)

Москва, 2023 г.

СОДЕРЖАНИЕ

1 Введение

1.1 Индивидуальное задание

Гео-информационная система. Приложение хост-подсистемы передает в SPE структуры OBJECT_X и OBJECT_Y. В структуре OBJECT_X в поле ключа хранится координата X объекта (64 бит), а в поле значения идентификатор объекта (64 бит). В структуре OBJECT_Y в поле ключа хранится идентификатор объекта (64 бит), а в поле значения координата Y объекта (64 бит). По запросу пользователя выдать все объекты, попадающие в прямоугольную область с координатами вурхнего левого угла (x1,y1) и нижнего правого угла (x2,y2).

2 Выполнение задания

Листинг 1 – код хост-подсистемы

```
#include <iostream>
#include <iterator>
#include <string>
#include <regex>
#include <sstream>
#include <fstream>
#include <ctime>
#include "host_main.h"

using namespace std;

#define TEST_USER_COUNT 1000
#define TEST_ROLE_COUNT 1000
#define TEST_IDX_COUNT 20

int main(int argc, char** argv)
{
  ofstream log("prac1.log"); //поток выводасообщений
  unsigned long long offs=0ull;
  gpc *gpc64_inst; //указатель накласс  gpc
  unsigned long long x1, x2, y1, y2, mode;

  //Инициализация gpc
  if (argc<2) {
    log<<"Использование: host_main путь< кфайлу  rawbinary>"<<endl;
    return -1;
  }

  //Захват ядра gpc изапись  sw_kernel
  gpc64_inst = new gpc();
  log<<"Открывается доступк  "<<gpc64_inst->gpc_dev_path<<endl;
  if (gpc64_inst->load_swk(argv[1])==0) {
    log<<"Программное ядрозагруженоизфайла    "<<argv[1]<<endl;
  }
  else {
    log<<"Ошибка загрузки sw_kernel файла << argv[1]"<<endl;
    return -1;
  }
```

```
39
40  cout << "Введите 0, чтобыпрограммаработаласоструктуройпользователей      ,
       анесоструктурой    , заданнойповарианту  " << endl;
41  cin >> mode;
42
43  if (mode == 0) {
44    //Инициализация таблицыдлявложенногозапроса
45    gpc64_inst->start(__event__(update)); //обработчик вставки
46
47    regex select_regex_query("select +(.*?) +from +(.*?) +where
         +(.*?)=(.*?) +and +(.*?)>(.*);", //запрос
48          std::regex_constants::ECMAScript | std::regex_constants
               ::icase);
49
50    //й1- вариант: пересылкакороткихсообщений
51    for (uint32_t user=0;user<TEST_USER_COUNT;user++) {
52      for (uint32_t idx=0;idx<TEST_ROLE_COUNT;idx++,offs+=2) {
53        gpc64_inst->mq_send(users::key{.idx=idx,.user=user}); //
             запись ороли  #idx
54        gpc64_inst->mq_send(users::val{.role=idx,.time=time_t(0)});
             //роль ивремядоступа
55      }
56    }
57
58    //й2- вариант: блочнаяпередача
59    unsigned long long *buf = (unsigned long long*)malloc(sizeof(
         unsigned long long)*TEST_USER_COUNT*TEST_ROLE_COUNT*2);
60    for (uint32_t user=0,offs=0;user<TEST_USER_COUNT;user++) {
61      for (uint32_t idx=0;idx<TEST_ROLE_COUNT;idx++,offs+=2) {
62        buf[offs]=users::key{.idx=idx,.user=user};
63        buf[offs+1]=users::val{.role=idx,.time=time_t(idx*3600)};
64      }
65    }
66    auto send_buf_th = gpc64_inst->mq_send(sizeof(unsigned long
         long)*TEST_USER_COUNT*TEST_ROLE_COUNT*2,(char*)buf);
67    send_buf_th->join();
68    free(buf);
69    //Терминальный символ
70    gpc64_inst->mq_send(-1ull);
71
72    gpc64_inst->start(__event__(select)); //обработчик запросапоиска
73    while(1) {
```

```
74        string query1;
75        //разбор полейзапроса
76        smatch match_query1;
77        getline(cin, query1);
78        log<<"Введен запрос: "<<query1<<endl;
79        if (!query1.compare("exit")) {
80          gpc64_inst->mq_send(-1ull);
81          break;
82        }
83        if (regex_match (query1, match_query1, select_regex_query) &&
84          match_query1[3]=="user" &&
85          match_query1[5] == "time") {
86          //match_query1[1] - возвращаемоеполезапроса
87          //match_query1[2] - номерструктурызапроса
88          //match_query1[3] - полепоиска  1
89          //match_query1[4] - значениеполяпоиска   1
90          //match_query1[5] - полепоиска  2
91          //match_query1[6] - значениеполяпоиска   2
92          log << "Запрос принятвобработку  ." << endl;
93          log << "Поиск ролейпользователя  " << match_query1[4] << "и
               time > " << time_t(stoi(match_query1[6])) << endl;
94          gpc64_inst->mq_send(stoi(match_query1[4])); //пользователь
95          gpc64_inst->mq_send(stoi(match_query1[6])); //время доступа
96          while (1) {
97            uint64_t result = gpc64_inst->mq_receive();
98            if (result!=-1ull) {
99              cout << "Роль: " << users::val::from_int(result).role
                   << " - ";
100             cout << "Время доступа: " << users::val::from_int(result)
                   .time << endl;
101           } else {
102             break;
103           }
104         }
105       } else {
106         log << "Ошибка взапросе !" << endl;
107       }
108     }
109   } else {
110     log << "Инициализация начальныхзначений ...";
111
112     //Инициализация начальныхзначений
```

```
113     gpc64_inst->start(__event__(update_obj_x)); //обработчик вставки
114
115
116     //й1- вариант: пересылкакороткихсообщений
117     for (uint64_t idx=0; idx<TEST_IDX_COUNT; idx++) {
118         uint64_t x_coord = idx;
119         log << "Запись: x = " << x_coord << " idx = " << idx << endl
                ;
120         gpc64_inst->mq_send(objects_x::key{.x_coord=x_coord}); //
                запись ключа x_coord
121         gpc64_inst->mq_send(objects_x::val{.idx=idx}); //запись
                значения idx
122       }
123
124     //Терминальный символ
125     gpc64_inst->mq_send(-1ull);
126
127     gpc64_inst->start(__event__(update_obj_y)); //обработчик вставки
128
129     //й1- вариант: пересылкакороткихсообщений
130     for (uint64_t idx=0; idx<TEST_IDX_COUNT; idx++) {
131         uint64_t y_coord = (idx * 17) % 10;
132         log << "Запись: y = " << y_coord << " idx = " << idx << endl
                ;
133         gpc64_inst->mq_send(objects_y::key{.idx=idx}); //запись
                ключа idx
134         gpc64_inst->mq_send(objects_y::val{.y_coord=y_coord}); //
                запись значения y_coord
135       }
136
137     //Терминальный символ
138     gpc64_inst->mq_send(-1ull);
139
140     log << " завершена." << endl;
141
142     gpc64_inst->start(__event__(select_obj_xy)); //обработчик
           запросапоиска
143     while(1) {
144       cout << "Введите вершиныпрямоугольника  (x1, y1, x2, y2): ";
145       cin >> x1 >> y1 >> x2 >> y2;
146       //разбор полейзапроса
147       log << "Введены вершины: " << x1 << y1 << x2 << y2 << endl;
```

```
148         log << "Запрос принятвобработку  ." << endl;
149
150         gpc64_inst->mq_send(-2ull);
151
152         log << "mq_send(-2ull) completed" << endl;
153
154         uint64_t result = gpc64_inst->mq_receive();
155         log << "result = " << result << endl;
156
157         gpc64_inst->mq_send(x1);
158         gpc64_inst->mq_send(y1);
159         gpc64_inst->mq_send(x2);
160         gpc64_inst->mq_send(y2);
161
162         result = gpc64_inst->mq_receive();
163         if (result == -3ull)
164           log << "Вершины переданывобработку  ." << endl;
165
166         gpc64_inst->mq_send(-4ull);
167
168         cout << "Вершины, находящиесявнутризаданногопрямоугольника   :" <<
              endl;
169
170         while (1) {
171           uint64_t key = gpc64_inst->mq_receive();
172           if (key != -1ull) {
173             uint64_t val_x = gpc64_inst->mq_receive();
174             uint64_t val_y = gpc64_inst->mq_receive();
175
176             log << "индекс idx: " << key << endl;
177             log << "(x, y): " << val_x << ", " << val_y << endl;
178             cout << "(x, y): " << val_x << ", " << val_y << endl;
179           } else {
180             break;
181           }
182         }
183
184         // log << Поиск" ролейпользователя  " << match_query1[4] << и"
              time > " << time_t(stoi(match_query1[6])) << endl;
185         // gpc64_inst->mq_send(stoi(match_query1[4])); пользователь//
186         // gpc64_inst->mq_send(stoi(match_query1[6])); время// доступа
187         // while (1) {
```

```
188         //  uint64_t result = gpc64_inst->mq_receive();
189         //  if (result!=-1ull) {
190         //    cout << Роль": " << users::val::from_int(result).role
                << " - ";
191         //    cout << Время" доступа: " << users::val::from_int(result)
                .time << endl;
192         //  } else {
193         //    break;
194         //  }
195         // }
196         break;
197       }
198     }
199
200     log << "Выход!" << endl;
201     return 0;
202 }
```

Листинг 2 – код обработчика программного ядра

```
1  #include <stdlib.h>
2  #include <ctime>
3  #include <cmath>
4  #include "lnh64.hxx"
5  #include "gpc_io_swk.h"
6  #include "gpc_handlers.h"
7  #include "iterators.h"
8  #include "common_struct.h"
9  #include "compose_keys.hxx"
10
11 #define __fast_recall__
12
13 extern lnh lnh_core;
14 volatile unsigned int event_source;
15
16 int main(void) {
17         ////////////////////////////////////////////////////////
18         //                    Main Event Loop
19         ////////////////////////////////////////////////////////
20         //Leonhard driver structure should be initialised
21         lnh_init();
22         for (;;) {
```

```
23                  //Wait for event
24                  event_source = wait_event();
25                  switch(event_source) {
26                  ////////////////////////////////////////////
27                  //  Measure GPN operation frequency
28                  ////////////////////////////////////////////
29                  case __event__(update) : update(); break;
30                  case __event__(select) : select(); break;
31                  case __event__(update_obj_x) : update_obj_x();
                        break;
32                  case __event__(select_obj_x) : select_obj_x();
                        break;
33                  case __event__(update_obj_y) : update_obj_y();
                        break;
34                  case __event__(select_obj_xy) : select_obj_xy();
                        break;
35                  }
36                  set_gpc_state(READY);
37          }
38  }
39
40  //-------------------------------------------------------------
41  // Вставкаключаизначениявструктуру
42  //-------------------------------------------------------------
43
44  void update() {
45          while(1){
46                  users::key key=users::key::from_int(mq_receive());
47                  if (key==-1ull) break;
48                  users::val val=users::val::from_int(mq_receive());
49                  // Поляструктурымогутзаписыватьсявноследующимобразом
50                  //      auto new_key = users::key{.rec_idx=1,.user
                        =2};
51                  //      auto new_val = users::val{.role=3,.lst_time
                        =0}
52                  //
                        Копированиеполейвпеременныеможновыполнитьследующимобразом
                        :
53                  //      auto user = key.user;
54                  //      auto [lst_time,role] = val;
55                  USERS.ins_async(key,val); //Вставка
                        втаблицустипизацией     uint64_t
```

```
56          }
57  }
58
59  void update_obj_x() {
60          while(1){
61                  objects_x::key key=objects_x::key::from_int(
                        mq_receive());
62                  if (key==-1ull) break;
63                  objects_x::val val=objects_x::val::from_int(
                        mq_receive());
64                  OBJECTS_X.ins_async(key,val); //Вставка
                        втаблицустипизацией     uint64_t
65          }
66  }
67
68  void update_obj_y() {
69          while(1){
70                  objects_y::key key=objects_y::key::from_int(
                        mq_receive());
71                  if (key==-1ull) break;
72                  objects_y::val val=objects_y::val::from_int(
                        mq_receive());
73                  OBJECTS_Y.ins_async(key,val); //Вставка
                        втаблицустипизацией     uint64_t
74          }
75  }
76
77
78  //-------------------------------------------------------------
79  // Передатьвсеролипользователяивремядоступа
80  //-------------------------------------------------------------
81
82  void select() {
83          while(1){
84                  uint32_t quser = mq_receive(); // gpc64_inst->
                        mq_send(stoi(match_query1[4])); пользователь//
85                  if (quser==-1) break;
86                  uint32_t qtime = mq_receive(); // gpc64_inst->
                        mq_send(stoi(match_query1[6])); время// доступа
87                  //Найдем всеролипользователяипоследнеевремядоступа       :
88                  // Результатыпоискамогутбытьдоступныследующимобразом       :
```

```
89                  //      auto user = USERS.search(users::key{.idx
                       =1,.user=2}).key().user;
90                  //      auto role = USERS.search(users::key{.idx
                       =3,.user=4}).value().role;
91
92                  //Вариант 1 - обходзаписейпользователяявнымобразом
93                  auto crole = USERS.nsm(users::key{.idx=users::
                       idx_min,.user=quser});
94                  while (crole && crole.key().user==quser) {
95                          if (crole.value().time>qtime) mq_send(crole.
                               value());
96                          crole = USERS.nsm(crole.key());
97                  }
98
99                  //Вариант 2 - использованиеитератора
100                 // for (auto val : role_range(USERS,quser)) {
101                 //         if (val.time>qtime) mq_send(val);
102                 // }
103                 mq_send(-1ull);
104         }
105 }
106
107 void select_obj_x() {
108         uint64_t x1, y1, x2, y2;
109
110         while(1){
111                 uint64_t msg = mq_receive();
112
113                 if (msg == -2ull) {
114                         mq_send(-22ull);
115                         x1 = mq_receive();
116                         y1 = mq_receive();
117                         x2 = mq_receive();
118                         y2 = mq_receive();
119                         mq_send(-3ull);
120                 } else if (msg == -4ull) {
121                         //Вариант 1 - обходзаписейявнымобразом
122                         auto qidx = OBJECTS_X.nsm(objects_x::key{.
                                x_coord=objects_x::x_coord_max});
123                         while (qidx) {
124                                 mq_send(qidx.key());
125                                 mq_send(qidx.value());
```

```
126                                     qidx = OBJECTS_X.nsm(qidx.key());
127                             }
128                             mq_send(-1ull);
129                     }
130             }
131     }
132
133     void select_obj_xy() {
134             uint64_t x1, y1, x2, y2;
135
136             while(1){
137                     uint64_t msg = mq_receive();
138
139                     if (msg == -2ull) {
140                             mq_send(-22ull);
141                             x1 = mq_receive();
142                             y1 = mq_receive();
143                             x2 = mq_receive();
144                             y2 = mq_receive();
145                             mq_send(-3ull);
146                     } else if (msg == -4ull) {
147                             //Вариант 1 - обходзаписейявнымобразом
148                             auto q_obj_x = OBJECTS_X.nsm(objects_x::key
                                    {.x_coord=objects_x::x_coord_max});
149                             uint64_t qidx = q_obj_x.value();
150                             auto q_obj_y = OBJECTS_Y.search(objects_y::
                                    key{.idx=qidx});
151
152                             while (1)
153                             {
154                                     if (q_obj_x.key() >= std::min(x1,
                                            x2) && q_obj_x.key() <= std::max
                                            (x1, x2))
155                                             if (q_obj_y.value() >= std
                                                    ::min(x1, x2) && q_obj_y
                                                    .value() <= std::max(y1,
                                                     y2))
156                                             {
157                                                     mq_send(qidx);
158                                                     mq_send(q_obj_x.key
                                                            ());
```

```
159                                                     mq_send(q_obj_y.
                                                           value());
160                                             }
161
162                                     q_obj_x = OBJECTS_X.nsm(q_obj_x.key
                                           ());
163
164                                     if (!q_obj_x) break;
165                                     qidx = q_obj_x.value();
166                                     q_obj_y = OBJECTS_Y.search(
                                           objects_y::key{.idx=qidx});
167                             }
168
169                             mq_send(-1ull);
170                     }
171             }
172 }
```

3 Вывод

Была разработана хост-подсистема, а так же обработчик программного ядра. выполняющие индивидуальное заданиею Данные программы были протестированы на вычислитьсяном комплексе Тераграф.