Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Курсовая работа по курсу

«Операционные системы»

Группа: М80-206Б-20

Студент: Шипилова Т.П.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 08.01.24

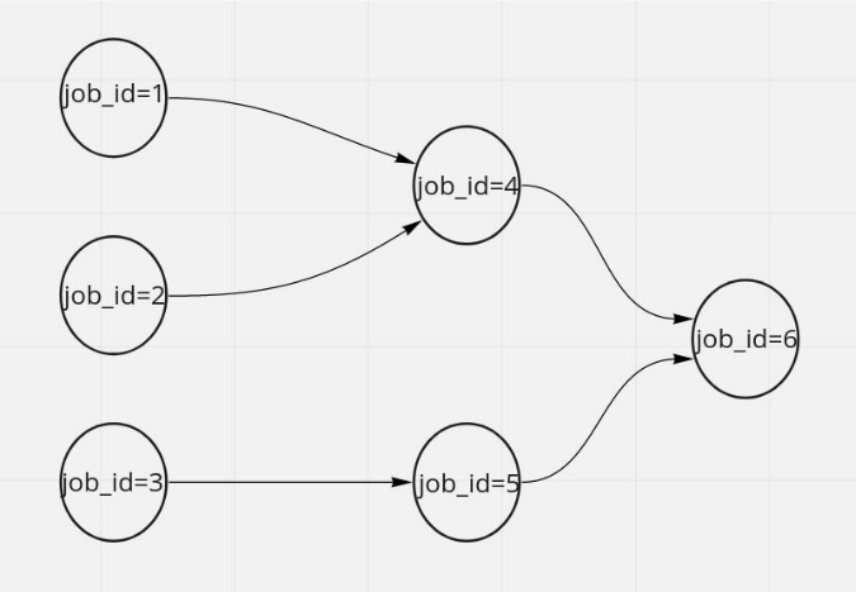
Москва, 2023

Постановка задачи

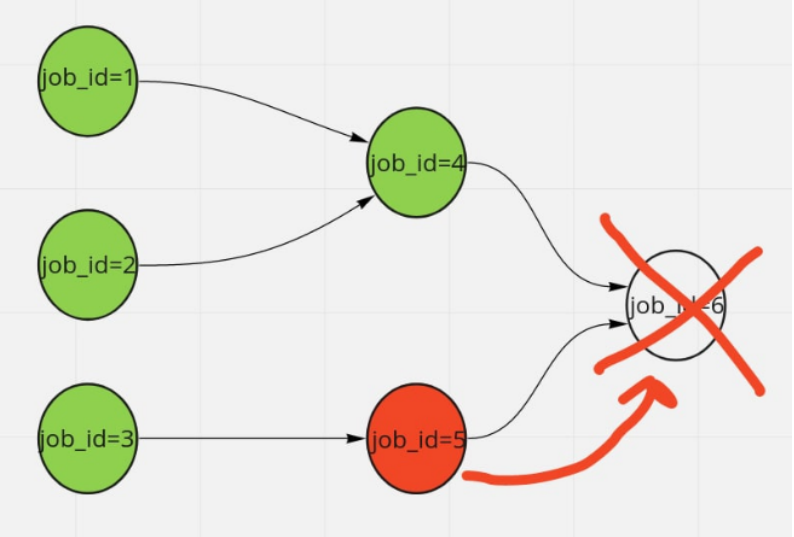
Вариант курсового проекта на 3.

На языке C\C++ написать программу, которая:

1. По конфигурационному файлу в формате yaml, json или ini принимает спроектированный DAG джобов и проверяет на корректность: отсутствие циклов, наличие только одной компоненты связанности, наличие стартовых и завершающих джоб. Структура описания джоб и их связей произвольная.



2. При завершении джобы с ошибкой, необходимо прервать выполнение всего DAG’а и всех запущенных джоб.



Общий метод и алгоритм решения

В данной работе происходит обработка конфигурационного файла и выполнение графа задач.

В функции main открывается конфигурационный файл "conf.ini".
Создаются пустые векторы startNodes и endNodes для хранения начальных и конечных узлов графа соответственно.
Создаются записи в map myF, где ключом является строковое название работы, а значением - указатель на соответствующую функцию.
Внутри цикла while считывается каждая строка из файла и проверяется на наличие символа '='. Если символ '=' найден, то строка разделяется на ключ и значение по символу '='.

В зависимости от ключа происходит определенное действие:

- Если ключ равен "NODE", создается новый узел с соответствующим идентификатором ID.
- Если ключ равен "CONNECTION", из значения извлекаются два числа (from и to), которые соответствуют связи между узлами графа.
- Если ключ равен "START_NODE", значение преобразуется в число и добавляется в вектор startNodes.
- Если ключ равен "END_NODE", значение преобразуется в число и добавляется в вектор endNodes.
- Если ключ равен "JOB", указатель job_node_now устанавливается на текущий узел в графе и сохраняется работа (value).

Вызывается функция all_job, передавая в нее стартовые узлы графа для выполнения задач. Внутри функции создается очередь queue и добавляются начальные узлы графа. Затем, для каждого узла в очереди выполняется следующее:

- Извлекается первый узел из очереди и получается указатель на него.
- Проверяется, есть ли узле задача для выполнения. Если есть, то вызывается соответствующая функция из myF и результат сохраняется в переменную res.
- Проверяется, является ли результат ошибкой (-1). Если да, то выводится сообщение об ошибке и цикл обработки прерывается.
- Переключается состояние узла на занятость (busy).
- Получается количество наследников узла и если оно не нулевое, то извлекается первый наследник и проверяется его состояние занятости.
- Если наследник свободен, то его состояние переключается на занятость и он добавляется в очередь.
- Обработка всех узлов в графе продолжается до тех пор, пока очередь не опустеет.

Код программы

dag.hpp

```
#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include <map>

#include <queue>

#include <set>

#include <algorithm>

#include <string>


using namespace std;


struct Node {

   vector<int> heirs;

   bool busy = false;

   string job = "not job";

};

typedef int (*FnPtr)();

map<int, Node> graph;

map<string, FnPtr> myF;


bool is_node_Cyclic(int v, bool visited[], bool in_rec_stack[]) {

   if (!visited[v]) {

       visited[v] = true;

       in_rec_stack[v] = true;


       for (int i = 0; i < graph[v].heirs.size(); i++) {
```

```
            int n = graph[v].heirs[i];
            if (!visited[n] && is_node_Cyclic(n, visited, in_rec_stack)) {
                return true;
            } else
            if (in_rec_stack[n]) {
                return true;
            }
        }

    }
    in_rec_stack[v] = false;
    return false;
}

bool is_Cyclic() {
    bool *visited = new bool[graph.size()];
    bool *in_rec_stack = new bool[graph.size()];
    for (int i = 0; i < graph.size(); i++) {
        visited[i] = false;
        in_rec_stack[i] = false;
    }

    for (int i = 0; i < graph.size(); i++) {
        if (is_node_Cyclic(i, visited, in_rec_stack)) {
            return true;
        }
    }
    return false;
}
```

```
bool is_Connected(std::vector<int>& endnods) {
   int count_out;
   int id_node;
   bool found;
   for (auto & pair : graph) {
       id_node = pair.first;
       count_out = pair.second.heirs.size();
       found = std::find(endnods.begin(), endnods.end(), id_node) !=
endnods.end();
       if (count_out > 1 && found == false)
           return false;
   }
   return true;
}



int FIRST() { return 1; }
int SECOND() { return 10; }
int THIRD() { return 11; }
int FORTH() {
   int i;
   cout << "Введите число: ";
   cin >> i;
   if(i == -1)
       return -1;
   else
       return i;
   }
int FIFTH() { return 101; }
```

```
int SIXTH() { return 110; }


typedef int (*FnPtr)();




void all_job(std::vector<int>& startNodes){
   queue<int> queue;
   int id_job_node, id_job_node_next, size;
   Node *job_node;
   string job;
   Node *next_job_node;
   int res;
   for (int i = 0; i < startNodes.size(); i++){
       queue.push(startNodes[i]);
   }
   for (int i = 0; i < graph.size() - 1 ; i++){
       id_job_node = queue.front();
       cout << "JOB: " << id_job_node << endl;
       queue.pop();
       job_node = &(graph.find(id_job_node) -> second);
       job = job_node->job;
       if(job != "not job"){
           res = myF[job]();
           if (res == -1){
               cout << "error" << endl;
               break;
           }
           cout << res << endl;
```

```
        }
        else{
            cout << job << endl;
        }
        job_node->busy = true;
        size = job_node->heirs.size();
        if(size != 0){
            id_job_node_next = job_node->heirs.front();
            next_job_node = &(graph.find(id_job_node_next) -> second);
            if(next_job_node->busy == false){
                next_job_node->busy = true;
                queue.push(id_job_node_next);
            }
        }
    }

}
```

main.cpp

```
#include <iostream>
#include <fstream>
#include "dag.hpp"

using namespace std;

int main() {
    ifstream configFile("conf.ini");
    vector<int> startNodes;
    vector<int> endNodes;
    Node *job_node_now;
```

```
int ID;

string line;

myF["FIRST"] = FIRST;

myF["SECOND"] = SECOND;

myF["THIRD"] = THIRD;

myF["FORTH"] = FORTH;

myF["FIFTH"] = FIFTH;

myF["SIXTH"] = SIXTH;

while (getline(configFile, line)) {

    if (line.find('=') != string::npos) {

        istringstream is(line);

        string key, value;

        if (getline(is, key, '=') && getline(is, value)) {

            if (key == "NODE") {

                Node node;

                ID = stoi(value);

                graph[stoi(value)] = node;

            } else

            if (key == "CONNECTION") {

                int from, to;

                istringstream(value) >> from >> to;

                graph[from].heirs.push_back(to);

            } else

            if (key == "START_NODE") {

                startNodes.push_back(stoi(value));

            } else

            if (key == "END_NODE") {

                endNodes.push_back(stoi(value));

            } else
```

```
                if (key == "JOB") {
                    job_node_now = &(graph.find(ID) -> second);
                    job_node_now->job = value;
                }
            }
        }
    }
    configFile.close();
    if (!is_Connected(endNodes)) {
        cout << "Граф не связан или пересвязан" << endl;
        return 1;
        }
    if (is_Cyclic()) {
        cout << "Граф содержит циклы" << endl;
        return 1;
        }
    if (startNodes.empty() || endNodes.empty()) {
        cout << "Отсутствуют стартовые или завершающие узлы" << endl;
        return 1;
        }
    all_job(startNodes);
    return 0;
}
```

conf.ini

```
[NODE]
NODE=1
JOB=FIRST
NODE=2
JOB=SECOND
```

```
NODE=3

JOB=THIRD

NODE=4

JOB=FORTH

NODE=5

JOB=FIFTH

NODE=6

JOB=SIXTH

[CONNECTION]

CONNECTION=1 4

CONNECTION=2 4

CONNECTION=3 5

CONNECTION=4 6

CONNECTION=5 6

[START_NODE]

START_NODE=1

START_NODE=2

START_NODE=3

[END_NODE]

END_NODE=6
```

Протокол работы программы

Тестирование:

```
./conf
JOB: 1
1
JOB: 2
10
JOB: 3
11
JOB: 4
Введите число: 5
5
JOB: 5
101
JOB: 6
110
```

Strace:

```
strace ./conf
execve("./conf", ["./conf"], 0x7ffe5d770790 /* 74 vars */) = 0
brk(NULL)                               = 0x55e40877a000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffec1536730) = -1 EINVAL (Недопустимый аргумент)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7469fee000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=71451, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 71451, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f7469fdc000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7469c00000
mprotect(0x7f7469c9a000, 1576960, PROT_NONE) = 0
mmap(0x7f7469c9a000, 1118208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x9a000) = 0x7f7469c9a000
mmap(0x7f7469dab000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1ab000) = 0x7f7469dab000
mmap(0x7f7469e1b000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x21a000) = 0x7f7469e1b000
```

```
     mmap(0x7f7469e29000, 10432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f7469e29000

     close(3)                                = 0

     openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3

     read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

     newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...}, AT_EMPTY_PATH) = 0

     mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7469fbc000

     mmap(0x7f7469fbf000, 94208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x3000) = 0x7f7469fbf000

     mmap(0x7f7469fd6000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000)
= 0x7f7469fd6000

     mmap(0x7f7469fda000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1d000) = 0x7f7469fda000

     close(3)                                = 0

     openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

     read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0"..., 832) =
832

     pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64)
= 784

     pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0"..., 48,
848) = 48

     pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0
=\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68

     newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, AT_EMPTY_PATH) = 0

     pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64)
= 784

     mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7469800000

     mmap(0x7f7469828000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7f7469828000

     mmap(0x7f74699bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1bd000) = 0x7f74699bd000

     mmap(0x7f7469a15000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x214000) = 0x7f7469a15000

     mmap(0x7f7469a1b000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f7469a1b000

     close(3)                                = 0

     openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

     read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

     newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) = 0

     mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7469ed5000

     mmap(0x7f7469ee3000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe000) = 0x7f7469ee3000
```

```
     mmap(0x7f7469f5f000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8a000) = 0x7f7469f5f000

     mmap(0x7f7469fba000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe4000) = 0x7f7469fba000

     close(3)                                = 0

     mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7469ed3000

     arch_prctl(ARCH_SET_FS, 0x7f7469ed43c0) = 0

     set_tid_address(0x7f7469ed4690)         = 13887

     set_robust_list(0x7f7469ed46a0, 24)     = 0

     rseq(0x7f7469ed4d60, 0x20, 0, 0x53053053) = 0

     mprotect(0x7f7469a15000, 16384, PROT_READ) = 0

     mprotect(0x7f7469fba000, 4096, PROT_READ) = 0

     mprotect(0x7f7469fda000, 4096, PROT_READ) = 0

     mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7469ed1000

     mprotect(0x7f7469e1b000, 45056, PROT_READ) = 0

     mprotect(0x55e407756000, 4096, PROT_READ) = 0

     mprotect(0x7f746a028000, 8192, PROT_READ) = 0

     prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

     munmap(0x7f7469fdc000, 71451)           = 0

     getrandom("\x06\x64\xb7\x35\xa9\xf9\x65\xf3", 8, GRND_NONBLOCK) = 8

     brk(NULL)                               = 0x55e40877a000

     brk(0x55e40879b000)                     = 0x55e40879b000

     futex(0x7f7469e2977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0

     openat(AT_FDCWD, "conf.ini", O_RDONLY)  = 3

     read(3, "[NODE]\nNODE=1\nJOB=FIRST\nNODE=2\nJ"..., 8191) = 271

     read(3, "", 8191)                       = 0

     close(3)                                = 0

     newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH)
= 0

     write(1, "JOB: 1\n", 7JOB: 1

     )                  = 7

     write(1, "1\n", 21

     )                  = 2

     write(1, "JOB: 2\n", 7JOB: 2

     )                  = 7

     write(1, "10\n", 310
```

```
)                   = 3

write(1, "JOB: 3\n", 7JOB: 3

)                   = 7

write(1, "11\n", 311

)                   = 3

write(1, "JOB: 4\n", 7JOB: 4

)                   = 7

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\321\207\320\270\321\201\320\273\320\276: ", 27Введите число: ) = 27

newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH)
= 0

read(0, 5

"5\n", 1024)                    = 2

write(1, "5\n", 25

)                   = 2

write(1, "JOB: 5\n", 7JOB: 5

)                   = 7

write(1, "101\n", 4101

)                   = 4

write(1, "JOB: 6\n", 7JOB: 6

)                   = 7

write(1, "110\n", 4110

)                   = 4

lseek(0, -1, SEEK_CUR)                = -1 ESPIPE (Недопустимая операция смещения)

exit_group(0)                         = ?

+++ exited with 0 +++
```

Вывод

В результате выполнения данной курсовой работы я узнала, что представляют собой файлы конфигурации ini, а также научилась с ними работать. Кроме того, я научилась создавать направленный ацикличный граф джобов и запускать эти джобы.