Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Курсовой проект по курсу**

**«Операционные системы»**

Группа: М8О-206Б-22

Студент: Носов А.К.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 06.03.2024

Москва, 2024

**Постановка задачи**

**Вариант 2 на 3.**

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

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.

При завершении джобы с ошибкой, необходимо прервать выполнение

всего DAG’а и всех запущенных джоб.

**Общий метод и алгоритм решения**

Программа состоит из нескольких частей. Сначала по конфигурационному файлу считывается информация. Затем проектируется DAG джобов. Потом проходимся по DAG и выполняем джобы.

**Код программы**

**main.cpp**

#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 work = "not job";

};

typedef int (\*FnPtr)();

map<int, Node> graph;

map<string, FnPtr> myF;

bool Cyclic\_one(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] && Cyclic\_one(n, visited, in\_rec\_stack)) {

return true;

} else if (in\_rec\_stack[n]) {

return true;

}

}

}

in\_rec\_stack[v] = false;

return false;

}

bool 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 (Cyclic\_one(i, visited, in\_rec\_stack)) {

return true;

}

}

return false;

}

bool 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 AAA() { return 111; }

int BBB() { return 222; }

int CCC() { return 333; }

int DDD() {

int i;

cout << "Введите число: ";

cin >> i;

if(i == -1)

return -1;

else

return i;

}

int EEE() { return 555; }

int FFF() { return 666; }

typedef int (\*FnPtr)();

void all\_work(std::vector<int>& startNodes){

queue<int> queue;

int id\_work\_node, id\_work\_node\_next, size;

Node \*work\_node;

string work;

Node \*next\_work\_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\_work\_node = queue.front();

cout << "JOB: " << id\_work\_node << endl;

queue.pop();

work\_node = &(graph.find(id\_work\_node) -> second);

work = work\_node->work;

if(work != "not job"){

res = myF[work]();

if (res == -1){

cout << "error" << endl;

break;

}

cout << res << endl;

}

else{

cout << work << endl;

}

work\_node->busy = true;

size = work\_node->heirs.size();

if(size != 0){

id\_work\_node\_next = work\_node->heirs.front();

next\_work\_node = &(graph.find(id\_work\_node\_next) -> second);

if(next\_work\_node->busy == false){

next\_work\_node->busy = true;

queue.push(id\_work\_node\_next);

}

}

}

}

int main() {

ifstream configFile("dag.ini");

string line;

vector<int> startNodes;

vector<int> endNodes;

int id;

Node \*work\_node\_now;

myF["AAA"] = AAA;

myF["BBB"] = BBB;

myF["CCC"] = CCC;

myF["DDD"] = DDD;

myF["EEE"] = EEE;

myF["FFF"] = FFF;

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") {

work\_node\_now = &(graph.find(id) -> second);

work\_node\_now->work = value;

}

}

}

}

configFile.close();

if (!Connected(endNodes)) {

cout << "Граф не связан или пересвязан" << endl;return 1;}

if (Cyclic()) {

cout << "Граф содержит циклы" << endl;return 1;}

if (startNodes.empty() || endNodes.empty()) {

cout << "Отсутствуют стартовые или завершающие узлы" << endl; return 1;}

all\_work(startNodes);

return 0;

}

**Протокол работы программы**

**Тестирование:**

alex@alex-HP-ENVY-x360-Convertible-13-ay1xxx:~/Desktop/OSLabs/KP$ ./a.out

JOB: 1

111

JOB: 2

222

JOB: 3

333

JOB: 4

Введите число: 5

5

JOB: 5

555

JOB: 6

666

**Strace:**

strace ./a.out

execve("./a.out", ["./a.out"], 0x7ffe60eab5c0 /\* 87 vars \*/) = 0

brk(NULL) = 0x55839fe4d000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffe966f8cb0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe98a1d9000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=73283, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 73283, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fe98a1c7000

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"..., 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) = 0x7fe989e00000

mprotect(0x7fe989e9a000, 1576960, PROT\_NONE) = 0

mmap(0x7fe989e9a000, 1118208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9a000) = 0x7fe989e9a000

mmap(0x7fe989fab000, 454656, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ab000) = 0x7fe989fab000

mmap(0x7fe98a01b000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x21a000) = 0x7fe98a01b000

mmap(0x7fe98a029000, 10432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fe98a029000

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\0\3\0>\0\1\0\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) = 0x7fe98a1a7000

mmap(0x7fe98a1aa000, 94208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7fe98a1aa000

mmap(0x7fe98a1c1000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a000) = 0x7fe98a1c1000

mmap(0x7fe98a1c5000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7fe98a1c5000

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\0\3\0>\0\1\0\0\0P\237\2\0\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\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2220400, ...}, 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, 2264656, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fe989a00000

mprotect(0x7fe989a28000, 2023424, PROT\_NONE) = 0

mmap(0x7fe989a28000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fe989a28000

mmap(0x7fe989bbd000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7fe989bbd000

mmap(0x7fe989c16000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7fe989c16000

mmap(0x7fe989c1c000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fe989c1c000

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\0\3\0>\0\1\0\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) = 0x7fe98a0c0000

mmap(0x7fe98a0ce000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7fe98a0ce000

mmap(0x7fe98a14a000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x8a000) = 0x7fe98a14a000

mmap(0x7fe98a1a5000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe4000) = 0x7fe98a1a5000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe98a0be000

arch\_prctl(ARCH\_SET\_FS, 0x7fe98a0bf3c0) = 0

set\_tid\_address(0x7fe98a0bf690) = 26839

set\_robust\_list(0x7fe98a0bf6a0, 24) = 0

rseq(0x7fe98a0bfd60, 0x20, 0, 0x53053053) = 0

mprotect(0x7fe989c16000, 16384, PROT\_READ) = 0

mprotect(0x7fe98a1a5000, 4096, PROT\_READ) = 0

mprotect(0x7fe98a1c5000, 4096, PROT\_READ) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe98a0bc000

mprotect(0x7fe98a01b000, 45056, PROT\_READ) = 0

mprotect(0x55839e1cc000, 4096, PROT\_READ) = 0

mprotect(0x7fe98a213000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7fe98a1c7000, 73283) = 0

getrandom("\xa2\x42\xe1\x3c\xf5\x65\xaf\xc9", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55839fe4d000

brk(0x55839fe6e000) = 0x55839fe6e000

futex(0x7fe98a02977c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

openat(AT\_FDCWD, "dag.ini", O\_RDONLY) = 3

read(3, "[START\_NODE]\nSTART\_NODE=1\nSTART\_"..., 8191) = 258

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, "111\n", 4111

) = 4

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

) = 7

write(1, "222\n", 4222

) = 4

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

) = 7

write(1, "333\n", 4333

) = 4

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, "555\n", 4555

) = 4

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

) = 7

write(1, "666\n", 4666

) = 4

lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

В результате выполнения этого практического задания я узнал, что такое файлы конфигурации ini, научился с ними работать. Кроме того, научился создавать Dag джобов. Запускать эти джобы.