

Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу
«Операционные системы»**

Студент: Матвеев Д. Е.
Группа: М8О-207Б-21
Вариант: 6
Преподаватель: Черемисинов Максим
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Замеры времени
8. Выводы

Репозиторий

<https://github.com/MrDenli/OsLabs>

Постановка задачи

Цель работы

Приобретение практических навыков в:

Управление потоками в ОС

Обеспечение синхронизации между потоками

Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При

обработки использовать стандартные средства создания потоков операционной системы

(Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Получившиеся результаты необходимо объяснить.

б) Произвести перемножение 2-ух матриц, содержащих комплексные числа

Общие сведения о программе

Программа компилируется из файла main.c при помощи make. В программе реализована многопоточность. Функции для работы с потоками, которые я использовал:

- `pthread_create()` — создание потока с передачей ему аргументов. В случае успеха возвращает 0.
- `pthread_join()` — ожидает завершения потока обозначенного `THREAD_ID`. Если этот поток к тому времени был уже завершен, то функция немедленно возвращает значение.
- `pthread_mutex_init()` — инициализация мьютекса
- `pthread_mutex_lock()` — блокировка мьютекса

- pthread_mutex_lock() — открытие доступа к мьютексу
- pthread_mutex_destroy() - удаление мьютекса

Общий метод и алгоритм решения:

Сначала нам нужно научиться работать с комплексными числами для этого пишем структуру complex_number, функции умножения комплексных чисел, сложения комплексных чисел, считывания и вывода комплексных чисел.

Дальше задача становится достаточно тривиальной. Основная идея заключается в том, что мы делим количество строчек получившейся матрицы на количество потоков, столько строчек будет обрабатывать один поток. Запускаем эти потоки, каждый поток обрабатывает свое количество строчек матрицы.

Исходный код

```

1. #include<pthread.h>
2. #include<iostream>
3. #include<ctime>
4. #include<vector>
5. #include<fstream>
6. #include<chrono>
7.
8. using namespace std;
9.
10. pthread_mutex_t mutex;
11. int flag = 0;
12. int cell = 0;
13.
14. struct complex_number {
15. int a;
16. int b;
17. };
18.
19. vector<vector<complex_number>> answer;
20.
21. complex_number complex_multiplication(complex_number X, complex_number Y) {
22. complex_number ans;
23. ans.a = X.a * Y.a + (-1) * X.b * Y.b;
24. ans.b = X.a * Y.b + X.b * Y.a;
25. return ans;
26. }
27.
28. complex_number complex_adition(complex_number X, complex_number Y) {
29. complex_number ans;
30. ans.a = X.a + Y.a;
31. ans.b = X.b + Y.b;
32. return ans;
33. }

```

```

34.
35. void write_complex_number(complex_number X) {
36.     cout << X.a;
37.     if (X.b == 0) {
38.         return;
39.     }
40.     if (X.b > 0) {
41.         cout << "+";
42.     }
43.     cout << X.b << "i";
44.     return;
45. }
46.
47. complex_number read_complex_number() {
48.     complex_number ans;
49.     cin >> ans.a;
50.     cin >> ans.b;
51.     char temp;
52.     cin >> temp;
53.     return ans;
54. }
55.
56. struct arg_to_thread {
57.     vector<vector<complex_number>> A;
58.     vector<vector<complex_number>> B;
59.     int partition;
60.     int num_of_thread;
61.     int count_threads;
62.     int n1;
63.     int m1;
64.     int n2;
65.     int m2;
66.     int n_ans;
67.     int m_ans;
68. };
69.
70. void* thread_func(void *args)
71. {
72.     arg_to_thread* arguments = (arg_to_thread*) args;
73.     int num_of_thread = arguments->num_of_thread;
74.     int partition = arguments->partition;
75.     flag = 1;
76.     int count_threads = arguments->count_threads;
77.     int n1 = arguments->n1;
78.     int m1 = arguments->m1;
79.     int n2 = arguments->n2;
80.     int m2 = arguments->m2;
81.     int n_ans = arguments->n_ans;
82.     int m_ans = arguments->m_ans;
83.     for (int i = 0; i < partition; i++) {
84.         int I = num_of_thread * partition + i;
85.         for (int J = 0; J < m_ans; J++) {
86.             for (int k = 0; k < m1; k++) {
87.                 complex_number temp = complex_multiplication((arguments->A)[I][k], (arguments->B)[k][J]);
88.                 answer[I][J] = complex_adition(answer[I][J], temp);
89.             }
90.             cell++;
91.         }
92.     }
93.     return 0;
94. }
95.
96. int main(int argc, char const *argv[])
97. {
98.     string command;
99.     int n1, n2, m1, m2;
100.     vector<vector<complex_number>> A;
101.     vector<vector<complex_number>> B;

```

```

102. cout << "1) Напишите work, если хотите переумножить матрицы" << endl << "2) Введите
    test, если хотите провести тест скорости" << endl;
103. cin >> command;
104. if (command == "work") {
105.     cout << "Ведите размер первой матрицы" << endl;
106.     cin >> n1 >> m1;
107.     A.resize(n1, vector<complex_number>(m1));
108.     cout << "Ведите первую матрицу" << endl;
109.     for (int i = 0; i < n1; i++) {
110.         for (int j = 0; j < m1; j++) {
111.             A[i][j] = read_complex_number();
112.         }
113.     }
114.     cout << "Ведите размер второй матрицы" << endl;
115.     cin >> n2 >> m2;
116.     B.resize(n2, vector<complex_number>(m2,complex_number{}));
117.     cout << "Введите вторую матрицу" << endl;
118.     for (int i = 0; i < n2; i++) {
119.         for (int j = 0; j < m2; j++) {
120.             B[i][j] = read_complex_number();
121.         }
122.     }
123. }
124. else if (command == "test"){
125.     cout << "Ведите размер первой матрицы" << endl;
126.     cin >> n1 >> m1;
127.     cout << "Ведите размер второй матрицы" << endl;
128.     cin >> n2 >> m2;
129.     A.resize(n1, vector<complex_number>(m1,complex_number{}));
130.     B.resize(n2, vector<complex_number>(m2,complex_number{}));
131. }
132. if (m1 != n2) {
133.     cout << "Матрицы таких размеров нельзя перемножить" << endl;
134.     return 0;
135. }
136.
137. int n_ans = n1;
138. int m_ans = m2;
139. answer.resize(n1);
140. for (int i = 0; i < n1; i++) {
141.     answer[i].resize(m2);
142. }
143.
144. for (int i = 0; i < n1; i++) {
145.     for (int j = 0; j < m2; j++) {
146.         answer[i][j].a = 0;
147.         answer[i][j].b = 0;
148.     }
149. }
150.
151. cout << "введите количество потоков" << endl;
152. int count_threads;
153. cin >> count_threads;
154.
155. pthread_t threads[count_threads];
156. pthread_mutex_init(&mutex, NULL);
157. int partition = n_ans / count_threads;
158.
159. if (count_threads > n_ans*m_ans) {
160.     cout << "Введено слишком много потоков" << endl;
161.     count_threads = n_ans;
162. }
163.
164. struct arg_to_thread arg;
165. arg.partition = partition;
166. arg.count_threads = count_threads;
167. arg.n1 = n1;
168. arg.m1 = m1;

```

```

169. arg.n2 = n2;
170. arg.m2 = m2;
171. arg.n_ans = n_ans;
172. arg.m_ans = m_ans;
173.
174. arg.A.resize(n1, vector<complex_number>(m1));
175. arg.B.resize(n2, vector<complex_number>(m2));
176.
177. for (int i = 0; i < n1; i++) {
178.     for (int j = 0; j < m1; j++) {
179.         arg.A[i][j].a = A[i][j].a;
180.         arg.A[i][j].b = A[i][j].b;
181.     }
182. }
183.
184. for (int i = 0; i < n2; i++) {
185.     for (int j = 0; j < m2; j++) {
186.         arg.B[i][j].a = B[i][j].a;
187.         arg.B[i][j].b = B[i][j].b;
188.     }
189. }
190.
191. chrono::high_resolution_clock::time_point begin = chrono::high_resolution_clock::now();
192. for (int i = 0; i < count_threads; i++) {
193.     arg.num_of_thread = i;
194.     if (i == count_threads - 1) {
195.         partition += (n1 * m2 % count_threads);
196.         arg.partition = partition;
197.     }
198.     int status = pthread_create(&threads[i], NULL, thread_func, (void*)&arg);
199.     flag = 0;
200.     if (status != 0) {
201.         cout<<"Create thread error"<<endl;
202.     }
203. }
204.
205. for (int i = 0; i < count_threads; ++i) {
206.     pthread_join(threads[i], NULL);
207. }
208. chrono::high_resolution_clock::time_point end = chrono::high_resolution_clock::now();
209.
210. pthread_mutex_destroy(&mutex);
211. if (command == "work"){
212.     cout << "Получившаяся матрица:" << endl;
213.     for (int i = 0; i < n_ans; i++) {
214.         for (int j = 0; j < m_ans; j++) {
215.             write_complex_number(answer[i][j]);
216.             cout << " ";
217.         }
218.         cout << endl;
219.     }
220. }
221. cout<< "программа выполнена за :" << chrono::duration_cast<chrono::milliseconds>(end-
    begin).count() << " миллисекунды" << endl;
222. return 0;
223.
224. }

```

Демонстрация работы программы

```
Обзор Терминал Сб, 5 ноября 09:36 en
danil@MrDenli: ~/OCI/Lab-3

danil@MrDenli:~/OCI/Lab-3$ cmake --build .
Consolidate compiler generated dependencies of target main
[ 50%] Building CXX object CMakeFiles/main.dir/src/main.cpp.o
[100%] Linking CXX executable main
[100%] Built target main
danil@MrDenli:~/OCI/Lab-3$ ./main
Введите количество потоков:
2
Ведите размер первой матрицы
4 4
Ведите первую матрицу
5+7i 6-6i 6-7i 12+6i
6+4i 8-3i -3-7i 76+5i
6-7i 4+54i 10-20i 43-3i
5-5i 7+8i 4-3i -1-8i
Ведите размер второй матрицы
4 5
Введите вторую матрицу
5+7i 6-6i 6-7i 12+6i 5+5i
6+4i 8-3i -3-7i 76+5i 6-5i
6-7i 4+54i 10-20i 43-3i 1-2i
5-5i 7+8i 4-3i -1-8i -1-10i
Получившаяся матрица:
113-56i 540+380i 5-219i 777-733i 36-151i
400-300i 973+393i 168-283i 485-1009i 0-774i
7-73i 1633+1125i 216-815i 451+2845i 256-168i
28+5i 315+123i -18-277i 682+488i 51+20i
370
danil@MrDenli:~/OCI/Lab-3$
```

Замеры времени

	1 поток	2 потока	4 потока	16 потоков
Матрицы 100x200 и 200x100	47 мс	29мс	19мс	15мс
Матрицы 1000x1000 и 1000x1000	25971мс	17144мс	5809мс	2906мс

Выводы

Мне понравилась данная лабораторная работа, я научился работать с потоками на языке с++. Я считаю эта лабораторная работа очень полезна для меня, потому

что полученные навыки с большой вероятностью помогут мне в будущем. Работа с потоками незаменимый навык для работы программистом, а поняв, как работают потоки на языке c++, можно легко научиться работать с потоками на других языках.