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: Боев Савелий Сергеевич

Группа: М8О-207Б-21

Вариант: 11

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

[Репозиторий 3](#_Toc123904682)

[Постановка задачи 3](#_Toc123904683)

[Цель работы 3](#_Toc123904684)

[Задание 3](#_Toc123904685)

[Исходный код 4](#_Toc123904686)

[Демонстрация работы программы 9](#_Toc123904687)

[Выводы 10](#_Toc123904688)

# **Репозиторий**

https://github.com/IamNoobLEL/Labs-OSi

# **Постановка задачи**

## **Цель работы**

Изучение операционных систем

## **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.

Наложить K раз фильтры эрозии и наращивания на матрицу, состоящую из вещественных чисел. На выходе получается 2 результирующие матрицы.

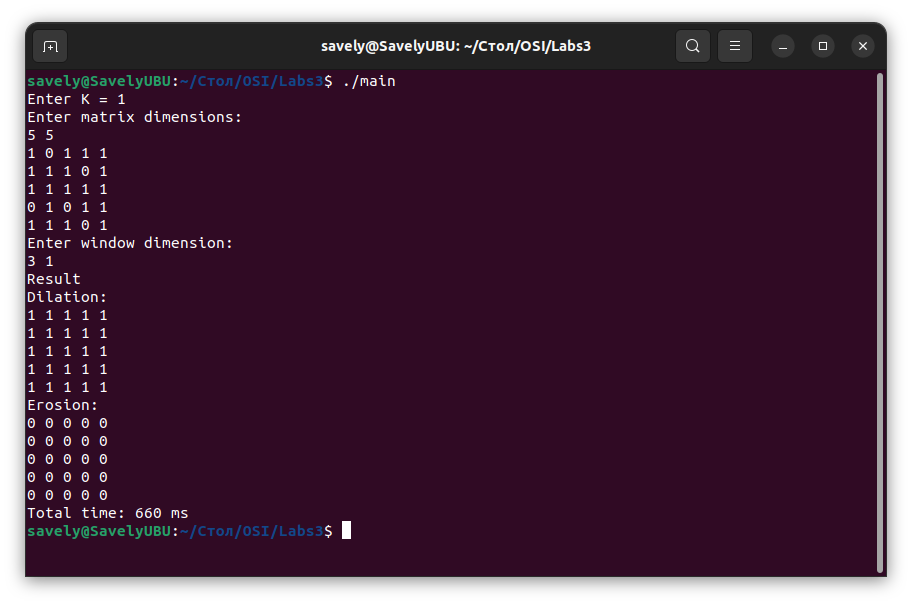
# **Исходный код**

1. #include <stdio.h>
2. #include <ctype.h>
3. #include <stdlib.h>
4. #include <stdbool.h>
5. #include <unistd.h>
6. #include <pthread.h>
7. #include <time.h>
8. #include <dirent.h>
9. #include <sys/time.h>

12. void print\_usage(char\* cmd) {
13. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Usage: %s [-threads num]**\n**", cmd);
14. }
16. bool read\_matrix(float\* matrix, size\_t rows, size\_t cols) {
17. for (size\_t i = 0; i < rows; i++) {
18. for (size\_t j = 0; j < cols; j++) {
19. if ([scanf](http://www.opengroup.org/onlinepubs/009695399/functions/scanf.html)("%f", &matrix[i \* cols + j]) != 1) {
20. [perror](http://www.opengroup.org/onlinepubs/009695399/functions/perror.html)("Error while reading matrix");
21. return **false**;
22. }
23. *//scanf("%f", &matrix[i\*cols + j]);*
24. *//printf("=== matrix[%ld] = %f\n", i\*cols + j, matrix[i\*cols + j]);*
25. }
26. }
27. return **true**;;
28. }
30. bool print\_matrix(float\* matrix, size\_t rows, size\_t cols) {
31. for (size\_t i = 0; i < rows; i++) {
32. for (size\_t j = 0; j < cols; j++) {
33. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%.20g ", matrix[i \* cols + j]);
34. }
35. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**");
36. }
37. return **false**;
38. }
40. void copy\_matrix(float\* from, float\* to, size\_t rows, size\_t cols) {
41. for (size\_t i = 0; i < rows; i++) {
42. for (size\_t j = 0; j < cols; j++) {
43. to[i \* cols + j] = from[i \* cols + j];
44. }
45. }
46. }
48. typedef struct {
49. int thread\_num;
50. int th\_count;
51. int rows;
52. int cols;
53. int w\_dim;
54. float\*\* matrix1;
55. float\*\* result1;
56. float\*\* matrix2;
57. float\*\* result2;
58. } thread\_arg;
60. void\* edit\_line(void\* argument) {
61. thread\_arg\* args = (thread\_arg\*)argument;
62. const int thread\_num = args->thread\_num;
63. const int th\_count = args->th\_count;
65. const int rows = args->rows;
66. const int cols = args->cols;
67. int offset = args->w\_dim / 2;
69. float\*\* matrix1\_ptr = args->matrix1;
70. float\*\* matrix2\_ptr = args->matrix2;
71. float\*\* result1\_ptr = args->result1;
72. float\*\* result2\_ptr = args->result2;
74. const float\* matrix1 = \*matrix1\_ptr;
75. const float\* matrix2 = \*matrix2\_ptr;
76. float\* result1 = \*result1\_ptr;
77. float\* result2 = \*result2\_ptr;
79. *//printf("\n=== IN THREAD %d ===\n", thread\_num);*
80. *// printf("offset = %d\n", offset);*
82. for (int th\_row = thread\_num; th\_row < rows; th\_row += th\_count) {
83. *//printf("THREAD %d ROW %d\n", thread\_num, th\_row);*
84. for (int th\_col = 0; th\_col < cols; th\_col++) {
85. *// printf(" th\_col = %d\n", th\_col);*
86. float max = matrix1[th\_row \* cols + th\_col];
87. float min = matrix2[th\_row \* cols + th\_col];
88. for (int i = th\_row - offset; i < th\_row + offset + 1; i++) {
89. for (int j = th\_col - offset; j < th\_col + offset + 1; j++) {
90. float curr1, curr2;
91. if ((i < 0) || (i >= rows) || (j < 0) || (j >= cols)) {
92. curr1 = 0;
93. curr2 = 0;
94. } else {
95. curr1 = matrix1[i \* cols + j];
96. curr2 = matrix2[i \* cols + j];
97. }
98. *// printf("[%d][%d] ", i, j);*
99. if (curr1 > max) {
100. max = curr1;
101. }
102. if (curr2 < min) {
103. min = curr2;
104. }
105. }
106. *// printf("\n");*
107. }
109. result1[th\_row \* cols + th\_col] = max;
110. result2[th\_row \* cols + th\_col] = min;
111. }
112. *//printf("\n");*
113. }
114. pthread\_exit(NULL); *// Заканчиваем поток*
115. }
117. void put\_filters(float\*\* matrix\_ptr, size\_t rows, size\_t cols, size\_t w\_dim, float\*\* res1\_ptr, float\*\* res2\_ptr, int filter\_cnt, int th\_count) {
118. float\* tmp1 = (float\*)[malloc](http://www.opengroup.org/onlinepubs/009695399/functions/malloc.html)(rows \* cols \* sizeof(float));
119. if (!tmp1) {
120. [perror](http://www.opengroup.org/onlinepubs/009695399/functions/perror.html)("Error while allocating matrix**\n**");
121. [exit](http://www.opengroup.org/onlinepubs/009695399/functions/exit.html)(1);
122. }
123. float\*\* matrix1\_ptr = &tmp1;
124. float\* tmp2 = (float\*)[malloc](http://www.opengroup.org/onlinepubs/009695399/functions/malloc.html)(rows \* cols \* sizeof(float));
125. if (!tmp2) {
126. [perror](http://www.opengroup.org/onlinepubs/009695399/functions/perror.html)("Error while allocating matrix**\n**");
127. [exit](http://www.opengroup.org/onlinepubs/009695399/functions/exit.html)(1);
128. }
129. float\*\* matrix2\_ptr = &tmp2;
130. copy\_matrix(\*matrix\_ptr, tmp1, rows, cols);
131. copy\_matrix(\*matrix\_ptr, tmp2, rows, cols);
133. pthread\_t ids[th\_count];
134. thread\_arg args[th\_count];
136. for (int k = 0; k < filter\_cnt; k++) {
137. for (int i = 0; i < th\_count; i++) {
138. args[i].thread\_num = i;
139. args[i].th\_count = th\_count;
140. args[i].rows = rows;
141. args[i].cols = cols;
142. args[i].w\_dim = w\_dim;
143. args[i].matrix1 = matrix1\_ptr;
144. args[i].result1 = res1\_ptr;
145. args[i].matrix2 = matrix2\_ptr;
146. args[i].result2 = res2\_ptr;
148. if (pthread\_create(&ids[i], NULL, edit\_line, &args[i]) != 0) {
149. [perror](http://www.opengroup.org/onlinepubs/009695399/functions/perror.html)("Can't create a thread.**\n**");
150. }
151. }
153. for(int i = 0; i < th\_count; i++) {
154. if (pthread\_join(ids[i], NULL) != 0) {
155. [perror](http://www.opengroup.org/onlinepubs/009695399/functions/perror.html)("Can't wait for thread**\n**");
156. }
157. }
159. if (filter\_cnt > 1) {
160. float\*\* swap = res1\_ptr;
161. res1\_ptr = matrix1\_ptr;
162. matrix1\_ptr = swap;
164. swap = res2\_ptr;
165. res2\_ptr = matrix2\_ptr;
166. matrix2\_ptr = swap;
167. }
168. }
170. [free](http://www.opengroup.org/onlinepubs/009695399/functions/free.html)(tmp1);
171. [free](http://www.opengroup.org/onlinepubs/009695399/functions/free.html)(tmp2);
172. }
174. int main(int argc, char\* argv[]) {
175. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Enter K = ");
176. int threads;
177. [scanf](http://www.opengroup.org/onlinepubs/009695399/functions/scanf.html)("%d", &threads);
179. if (argc == 3) {
180. threads = [atoi](http://www.opengroup.org/onlinepubs/009695399/functions/atoi.html)(argv[2]);
181. } else if (argc != 1) {
182. print\_usage(argv[0]);
183. return 0;
184. }
186. int rows;
187. int cols;
188. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Enter matrix dimensions:**\n**");
189. [scanf](http://www.opengroup.org/onlinepubs/009695399/functions/scanf.html)("%d", &cols);
190. [scanf](http://www.opengroup.org/onlinepubs/009695399/functions/scanf.html)("%d", &rows);
191. float\* matrix = (float\*)[malloc](http://www.opengroup.org/onlinepubs/009695399/functions/malloc.html)(rows \* cols \* sizeof(float));
192. float\* res1 = (float\*)[malloc](http://www.opengroup.org/onlinepubs/009695399/functions/malloc.html)(rows \* cols \* sizeof(float));
193. float\* res2 = (float\*)[malloc](http://www.opengroup.org/onlinepubs/009695399/functions/malloc.html)(rows \* cols \* sizeof(float));
194. if (!matrix || !res1 || !res2) {
195. [perror](http://www.opengroup.org/onlinepubs/009695399/functions/perror.html)("Error while allocating matrix**\n**");
196. return 1;
197. }
198. read\_matrix(matrix, rows, cols);
200. int w\_dim;
201. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Enter window dimension:**\n**");
202. [scanf](http://www.opengroup.org/onlinepubs/009695399/functions/scanf.html)("%d", &w\_dim);
203. if (w\_dim % 2 == 0) {
204. [perror](http://www.opengroup.org/onlinepubs/009695399/functions/perror.html)("Window dimension must be an odd number**\n**");
205. return 1;
206. }
208. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Result **\n**");
209. int k;
210. [scanf](http://www.opengroup.org/onlinepubs/009695399/functions/scanf.html)("%d", &k);
212. struct timeval start, end;
213. gettimeofday(&start, NULL);
215. put\_filters(&matrix, rows, cols, w\_dim, &res1, &res2, k, threads);
217. gettimeofday(&end, NULL);
219. long sec = end.tv\_sec - start.tv\_sec;
220. long microsec = end.tv\_usec - start.tv\_usec;
221. if (microsec < 0) {
222. --sec;
223. microsec += 1000000;
224. }
225. long elapsed = sec\*1000000 + microsec;

228. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Dilation:**\n**");
229. print\_matrix(res1, rows, cols);
230. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Erosion:**\n**");
231. print\_matrix(res2, rows, cols);
232. [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Total time: %ld ms**\n**", elapsed);
234. [free](http://www.opengroup.org/onlinepubs/009695399/functions/free.html)(res1);
235. [free](http://www.opengroup.org/onlinepubs/009695399/functions/free.html)(res2);
236. [free](http://www.opengroup.org/onlinepubs/009695399/functions/free.html)(matrix);
237. return 0;
238. }

## **Демонстрация работы программы**



# **Выводы**

Мною было освоено написании программы, создающей и производящей вычисления в нескольких потоках, а также синхронизация данных между этими потоками. Это позволяет серьезно сократить время выполнения некоторых задач, ибо действия выполняются не последовательно, а параллельно.