Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №1 по курсу**

**«Операционные системы»**

Группа: М8О-214Б-23

Студент: Гусев С.В.

Преподаватель: Бахарев В.Д.

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

Дата: 28.10.24

Москва, 2024

**Постановка задачи**

**Вариант 2.**

**Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, должно быть задано ключом запуска вашей программы.**

**Отсортировать массив целых чисел при помощи четно-нечетной сортировки Бетчера.**

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* int channel[2];

pipe(channel); – создает два канала связи.

* const pid\_t child = fork(); – создает дочерний процесс.
* pid\_t pid = getpid(); – получает номер текущего процесса.
* dup2(STDIN\_FILENO, channel[STDIN\_FILENO]); – перенаправляет стандартный ввод на дескриптор канала связи.
* int32\_t status = execv(path, args); – ● заменяет код новым программным кодом, указанным в path.
* wait(&child\_status); – родительский процесс ждет завершения дочернего процесса.

Решение:

1. Обрабатываю максимальное количество потоков переданное через аргументы командной строки.
2. С помощью функций написанных выше связываю родительский процесс с дочерним (передаю обработанный ввод).
3. В дочернем процессе считываю строку (используя read()) и находящиеся там символы записываю в массив типа int.
4. Создаю несколько потоков, в каждом из которых частично сортирую массив(в итоге получается полностью отсортированный т. к. во все потоки передается один и тот же массив).
5. Ответ вывожу в стандартный поток вывода (использую write()).

**Код программы**

**server.c**

1. #include <stdint.h>
2. #include <stdbool.h>
3. #include <unistd.h>
4. #include <sys/wait.h>
5. #include <stdlib.h>
6. #include <stdio.h>
7. static char CLIENT\_PROGRAM\_NAME[] = "client";
8. int main(int argc, char \*\*argv) {
9. if (argc == 1) {
10. char msg[1024];
11. uint32\_t len = snprintf(msg, sizeof(msg) - 1, "usage: %s threadNumber\n", argv[0]);
12. write(STDERR\_FILENO, msg, len);
13. exit(EXIT\_SUCCESS);
14. }
15. char progpath[1024];
16. {
17. ssize\_t len = readlink("/proc/self/exe", progpath, sizeof(progpath) - 1);
18. if (len == -1) {
19. const char msg[] = "error: failed to read full program path\n";
20. write(STDERR\_FILENO, msg, sizeof(msg));
21. exit(EXIT\_FAILURE);
22. }
23. while (progpath[len] != '/')
24. --len;
25. progpath[len] = '\0';
26. }
27. //
28. // char buf[4096];
29. // ssize\_t bytes;
30. int channel[2];
31. if (pipe(channel) == -1) {
32. const char msg[] = "error: failed to create pipe\n";
33. write(STDERR\_FILENO, msg, sizeof(msg));
34. exit(EXIT\_FAILURE);
35. }
36. const pid\_t child = fork();
37. switch (child) {
38. case -1: {
39. const char msg[] = "error: failed to spawn new process\n";
40. write(STDERR\_FILENO, msg, sizeof(msg));
41. exit(EXIT\_FAILURE);
42. } break;
43. case 0: {
44. pid\_t pid = getpid();
45. dup2(STDIN\_FILENO, channel[STDIN\_FILENO]);
46. // dup2(channel[STDIN\_FILENO], STDIN\_FILENO);
47. close(channel[STDOUT\_FILENO]);
48. {
49. char msg[64];
50. const int32\_t length = snprintf(msg, sizeof(msg), "%d: I'm a child\n", pid);
51. write(STDOUT\_FILENO, msg, length);
52. }
53. {
54. char path[1024];
55. snprintf(path, sizeof(path) - 1, "%s/%s", progpath, CLIENT\_PROGRAM\_NAME);
56. char \*const args[] = {CLIENT\_PROGRAM\_NAME, argv[1], NULL};
57. int32\_t status = execv(path, args);
58. if (status == -1) {
59. const char msg[] = "error: failed to exec into new executable image\n";
60. write(STDERR\_FILENO, msg, sizeof(msg));
61. exit(EXIT\_FAILURE);
62. }
63. }
64. } break;
65. default: {
66. pid\_t pid = getpid();
67. {
68. char msg[64];
69. const int32\_t length = snprintf(msg, sizeof(msg), "%d: I'm a parent, my child has PID %d\n", pid, child);
70. write(STDOUT\_FILENO, msg, length);
71. }
72. // while (bytes = read(STDIN\_FILENO, buf, sizeof(buf) - 1)) {
73. // write(channel[STDOUT\_FILENO], buf, bytes);
74. // }
75. int child\_status;
76. wait(&child\_status);
77. if (child\_status != EXIT\_SUCCESS) {
78. const char msg[] = "error: child exited with error\n";
79. write(STDERR\_FILENO, msg, sizeof(msg));
80. exit(child\_status);
81. }
82. } break;
83. }
84. }

**client.c**

1. #include <stdint.h>
2. #include <stdlib.h>
3. #include <unistd.h>
4. #include <fcntl.h>
5. #include <ctype.h>
6. #include <stdio.h>
7. #include <pthread.h>
8. typedef struct {
9. int num\_threads;
10. int \*array;
11. int ar\_len;
12. } thread\_data;
13. void swap(int \*a, int \*b) {
14. int tmp = \*a;
15. \*a = \*b;
16. \*b = tmp;
17. }
18. void\* BetcherSort(void\* arg) {
19. thread\_data \*data = (thread\_data \*)arg;
20. int num\_threads = data->num\_threads;
21. int \*array = data->array;
22. int len = data->ar\_len;
23. for (size\_t i = 0; i < (len / num\_threads + 1); i++) {
24. for (size\_t j = (i % 2) ? 0 : 1; j + 1 < len; j += 2) {
25. if (array[j] > array[j + 1]) {
26. swap(array + j, array + j + 1);
27. }
28. }
29. }
30. pthread\_exit(NULL);
31. return NULL;
32. }
33. int main(int argc, char \*\*argv) {
34. char buf[1];
35. char \*int\_in\_str = (char\*) malloc(sizeof (char));
36. if (int\_in\_str == NULL) {
37. const char msg[] = "error: failed to allocate memory\n";
38. write(STDERR\_FILENO, msg, sizeof(msg));
39. exit(EXIT\_FAILURE);
40. }
41. int int\_in\_str\_size = 1;
42. int ind\_str = 0;
43. ssize\_t bytes;
44. int \*array = (int\*) malloc(sizeof (int));
45. if (array == NULL) {
46. const char msg[] = "error: failed to allocate memory\n";
47. write(STDERR\_FILENO, msg, sizeof(msg));
48. exit(EXIT\_FAILURE);
49. }
50. int array\_size = 1;
51. int ind\_array = 0;
52. pid\_t pid = getpid();
53. {
54. char msg[128];
55. int32\_t len = snprintf(msg, sizeof(msg) - 1,
56. "%d: Start typing integers. Press 'Ctrl-D' or 'Enter' with no input to exit\n", pid);
57. write(STDOUT\_FILENO, msg, len);
58. }
59. while ((bytes = read(STDIN\_FILENO, buf, sizeof(buf))) != 0) {
60. if (bytes < 0) {
61. free(int\_in\_str);
62. free(array);
63. const char msg[] = "error: failed to read from stdin\n";
64. write(STDERR\_FILENO, msg, sizeof(msg));
65. exit(EXIT\_FAILURE);
66. }
67. if (isdigit(buf[0]) || (buf[0] == '-' && ind\_str == 0)) {
68. int\_in\_str[ind\_str] = buf[0];
69. ind\_str++;
70. if (ind\_str >= int\_in\_str\_size) {
71. char \*ptr;
72. int\_in\_str\_size \*= 2;
73. ptr = (char\*) realloc(int\_in\_str, int\_in\_str\_size \* sizeof(char));
74. if (ptr == NULL) {
75. free(int\_in\_str);
76. free(array);
77. const char msg[] = "error: failed to reallocate memory\n";
78. write(STDERR\_FILENO, msg, sizeof(msg));
79. exit(EXIT\_FAILURE);
80. }
81. int\_in\_str = ptr;
82. }
83. } else {
84. if (ind\_str == 0) {
85. continue;
86. }
87. int\_in\_str[ind\_str] = '\0';
88. array[ind\_array] = atoi(int\_in\_str);
89. ind\_array++;
90. if (ind\_array >= array\_size) {
91. array\_size \*= 2;
92. int \*ptr;
93. ptr = (int\*) realloc(array, array\_size \* sizeof(int));
94. if (ptr == NULL) {
95. free(array);
96. free(int\_in\_str);
97. const char msg[] = "error: failed to reallocate memory\n";
98. write(STDERR\_FILENO, msg, sizeof(msg));
99. exit(EXIT\_FAILURE);
100. }
101. array = ptr;
102. }
103. ind\_str = 0;
104. }
105. if (buf[0] == '\n' || buf[0] == EOF) {
106. break;
107. }
108. }
109. free(int\_in\_str);
110. int num\_of\_threads = atoi(argv[1]);
111. pthread\_t threads[num\_of\_threads];
112. thread\_data thread\_data\_array[num\_of\_threads];
113. for (int i = 0; i < num\_of\_threads; i++) {
114. thread\_data\_array[i].num\_threads = num\_of\_threads;
115. thread\_data\_array[i].array = array;
116. thread\_data\_array[i].ar\_len = ind\_array;
117. if (pthread\_create(&threads[i], NULL, BetcherSort, (void\*)&thread\_data\_array[i]) != 0) {
118. const char msg[] = "error: failed to create thread\n";
119. write(STDERR\_FILENO, msg, sizeof(msg));
120. exit(EXIT\_FAILURE);
121. }
122. }
123. for (int i = 0; i < num\_of\_threads; i++) {
124. pthread\_join(threads[i], NULL);
125. }
126. const char msg[] = "\nSorted array:\n";
127. write(STDOUT\_FILENO, msg, sizeof(msg));
128. char ans[20];
129. for (int i = 0; i < ind\_array; ++i) {
130. size\_t ansLen = snprintf(ans, sizeof(ans), ((i == (ind\_array - 1))? "%d\n" : "%d "), array[i]);
131. int32\_t written = write(STDOUT\_FILENO, ans, ansLen);
132. if (written != ansLen) {
133. const char msg[] = "error: failed to write to file\n";
134. write(STDERR\_FILENO, msg, sizeof(msg));
135. exit(EXIT\_FAILURE);
136. }
137. }
138. free(array);
139. return 0;
140. }

**Протокол работы программы**

**Тестирование:**

$ ./server 4

82178: I'm a parent, my child has PID 82179

82179: I'm a child

82179: Start typing integers. Press 'Ctrl-D' or 'Enter' with no input to exit

29 24 91 -40 -92 -95 86 -1 59 -12 98 41 35 80 70 -56 70 8 19 -29 -29 -6 90 -100 0 39 15 87 46 -1 -62

Sorted array:

-100 -95 -92 -62 -56 -40 -29 -29 -12 -6 -1 -1 0 8 15 19 24 29 35 39 41 46 59 70 70 80 86 87 90 91 98

You can use 'ps -T -p 82179' to see the threads of this process.

$ ./server 7

83019: I'm a parent, my child has PID 83020

83020: I'm a child

83020: Start typing integers. Press 'Ctrl-D' or 'Enter' with no input to exit

-99 9 -100 -24 60 -29 34 30 54 43 -6 96 -45 18 -1 -21 40 -43 9 -33 -58 9 88 -55 -14 -78 -44 46 28 59 -70 66 -77 -29 -46 58 -45 48 -27 53 67 5 -33 -82

Sorted array:

-100 -99 -82 -78 -77 -70 -58 -55 -46 -45 -45 -44 -43 -33 -33 -29 -29 -27 -24 -21 -14 -6 -1 5 9 9 9 18 28 30 34 40 43 46 48 53 54 58 59 60 66 67 88 96

You can use 'ps -T -p 83020' to see the threads of this process.

**Strace:**

$ strace -f ./server 4

execve("./server", ["./server", "4"], 0x7fff5f7d16d0 /\* 28 vars \*/) = 0

brk(NULL) = 0x55c5ab199000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fb1bc313000

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

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=20535, ...}) = 0

mmap(NULL, 20535, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fb1bc30d000

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\0\220\243\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

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 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, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fb1bc0fb000

mmap(0x7fb1bc123000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fb1bc123000

mmap(0x7fb1bc2ab000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1b0000) = 0x7fb1bc2ab000

mmap(0x7fb1bc2fa000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1fe000) = 0x7fb1bc2fa000

mmap(0x7fb1bc300000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fb1bc300000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fb1bc0f8000

arch\_prctl(ARCH\_SET\_FS, 0x7fb1bc0f8740) = 0

set\_tid\_address(0x7fb1bc0f8a10) = 83921

set\_robust\_list(0x7fb1bc0f8a20, 24) = 0

rseq(0x7fb1bc0f9060, 0x20, 0, 0x53053053) = 0

mprotect(0x7fb1bc2fa000, 16384, PROT\_READ) = 0

mprotect(0x55c5a57a4000, 4096, PROT\_READ) = 0

mprotect(0x7fb1bc34b000, 8192, PROT\_READ) = 0

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

munmap(0x7fb1bc30d000, 20535) = 0

readlink("/proc/self/exe", "/mnt/c/Users/\320\232\321\200\321\217/CLionProject"..., 1023) = 49

pipe2([3, 4], 0) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 83922 attached

, child\_tidptr=0x7fb1bc0f8a10) = 83922

[pid 83922] set\_robust\_list(0x7fb1bc0f8a20, 24 <unfinished ...>

[pid 83921] getpid( <unfinished ...>

[pid 83922] <... set\_robust\_list resumed>) = 0

[pid 83921] <... getpid resumed>) = 83921

[pid 83922] getpid( <unfinished ...>

[pid 83921] write(1, "83921: I'm a parent, my child ha"..., 44 <unfinished ...>

[pid 83922] <... getpid resumed>) = 83922

83921: I'm a parent, my child has PID 83922

[pid 83921] <... write resumed>) = 44

[pid 83922] dup2(0, 3 <unfinished ...>

[pid 83921] wait4(-1, <unfinished ...>

[pid 83922] <... dup2 resumed>) = 3

[pid 83922] close(4) = 0

[pid 83922] write(1, "83922: I'm a child\n", 1983922: I'm a child

) = 19

[pid 83922] execve("/mnt/c/Users/\320\232\321\200\321\217/CLionProjects/OSy/Lab2/client", ["client", "4"], 0x7ffc38aa5730 /\* 28 vars \*/) = 0

[pid 83922] brk(NULL) = 0x55fb1f25a000

[pid 83922] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f2c84e17000

[pid 83922] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 83922] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 4

[pid 83922] fstat(4, {st\_mode=S\_IFREG|0644, st\_size=20535, ...}) = 0

[pid 83922] mmap(NULL, 20535, PROT\_READ, MAP\_PRIVATE, 4, 0) = 0x7f2c84e11000

[pid 83922] close(4) = 0

[pid 83922] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 4

[pid 83922] read(4, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832

[pid 83922] pread64(4, "\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

[pid 83922] fstat(4, {st\_mode=S\_IFREG|0755, st\_size=2125328, ...}) = 0

[pid 83922] pread64(4, "\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

[pid 83922] mmap(NULL, 2170256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 4, 0) = 0x7f2c84bff000

[pid 83922] mmap(0x7f2c84c27000, 1605632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x28000) = 0x7f2c84c27000

[pid 83922] mmap(0x7f2c84daf000, 323584, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x1b0000) = 0x7f2c84daf000

[pid 83922] mmap(0x7f2c84dfe000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x1fe000) = 0x7f2c84dfe000

[pid 83922] mmap(0x7f2c84e04000, 52624, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f2c84e04000

[pid 83922] close(4) = 0

[pid 83922] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f2c84bfc000

[pid 83922] arch\_prctl(ARCH\_SET\_FS, 0x7f2c84bfc740) = 0

[pid 83922] set\_tid\_address(0x7f2c84bfca10) = 83922

[pid 83922] set\_robust\_list(0x7f2c84bfca20, 24) = 0

[pid 83922] rseq(0x7f2c84bfd060, 0x20, 0, 0x53053053) = 0

[pid 83922] mprotect(0x7f2c84dfe000, 16384, PROT\_READ) = 0

[pid 83922] mprotect(0x55faf05a5000, 4096, PROT\_READ) = 0

[pid 83922] mprotect(0x7f2c84e4f000, 8192, PROT\_READ) = 0

[pid 83922] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 83922] munmap(0x7f2c84e11000, 20535) = 0

[pid 83922] getrandom("\x32\x60\x1d\xfa\x42\xfb\xde\x11", 8, GRND\_NONBLOCK) = 8

[pid 83922] brk(NULL) = 0x55fb1f25a000

[pid 83922] brk(0x55fb1f27b000) = 0x55fb1f27b000

[pid 83922] getpid() = 83922

[pid 83922] write(1, "83922: Start typing integers. Pr"..., 7883922: Start typing integers. Press 'Ctrl-D' or 'Enter' with no input to exit

) = 78

[pid 83922] read(0, -68 70 37 -90 73 -35 -16 62 14 74 -4 -60 -65 30 51 35 95 37

"-", 1) = 1

[pid 83922] read(0, "6", 1) = 1

[pid 83922] read(0, "8", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "7", 1) = 1

[pid 83922] read(0, "0", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "3", 1) = 1

[pid 83922] read(0, "7", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "-", 1) = 1

[pid 83922] read(0, "9", 1) = 1

[pid 83922] read(0, "0", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "7", 1) = 1

[pid 83922] read(0, "3", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "-", 1) = 1

[pid 83922] read(0, "3", 1) = 1

[pid 83922] read(0, "5", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "-", 1) = 1

[pid 83922] read(0, "1", 1) = 1

[pid 83922] read(0, "6", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "6", 1) = 1

[pid 83922] read(0, "2", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "1", 1) = 1

[pid 83922] read(0, "4", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "7", 1) = 1

[pid 83922] read(0, "4", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "-", 1) = 1

[pid 83922] read(0, "4", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "-", 1) = 1

[pid 83922] read(0, "6", 1) = 1

[pid 83922] read(0, "0", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "-", 1) = 1

[pid 83922] read(0, "6", 1) = 1

[pid 83922] read(0, "5", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "3", 1) = 1

[pid 83922] read(0, "0", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "5", 1) = 1

[pid 83922] read(0, "1", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "3", 1) = 1

[pid 83922] read(0, "5", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "9", 1) = 1

[pid 83922] read(0, "5", 1) = 1

[pid 83922] read(0, " ", 1) = 1

[pid 83922] read(0, "3", 1) = 1

[pid 83922] read(0, "7", 1) = 1

[pid 83922] read(0, "\n", 1) = 1

[pid 83922] rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f2c84c98520, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f2c84c44320}, NULL, 8) = 0

[pid 83922] rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0

[pid 83922] mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f2c843fb000

[pid 83922] mprotect(0x7f2c843fc000, 8388608, PROT\_READ|PROT\_WRITE) = 0

[pid 83922] rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

[pid 83922] clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f2c84bfb990, parent\_tid=0x7f2c84bfb990, exit\_signal=0, stack=0x7f2c843fb000, stack\_size=0x7fff80, tls=0x7f2c84bfb6c0}strace: Process 83988 attached

<unfinished ...>

[pid 83988] rseq(0x7f2c84bfbfe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 83922] <... clone3 resumed> => {parent\_tid=[83988]}, 88) = 83988

[pid 83988] <... rseq resumed>) = 0

[pid 83922] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 83988] set\_robust\_list(0x7f2c84bfb9a0, 24 <unfinished ...>

[pid 83922] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 83988] <... set\_robust\_list resumed>) = 0

[pid 83922] mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 83988] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 83922] <... mmap resumed>) = 0x7f2c83bfa000

[pid 83988] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 83922] mprotect(0x7f2c83bfb000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 83988] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC <unfinished ...>

[pid 83922] <... mprotect resumed>) = 0

[pid 83988] <... openat resumed>) = 4

[pid 83922] futex(0x7f2c84e51a58, FUTEX\_WAIT\_PRIVATE, 2, NULL <unfinished ...>

[pid 83988] fstat(4, {st\_mode=S\_IFREG|0644, st\_size=20535, ...}) = 0

[pid 83988] mmap(NULL, 20535, PROT\_READ, MAP\_PRIVATE, 4, 0) = 0x7f2c84e11000

[pid 83988] close(4) = 0

[pid 83988] mmap(NULL, 134217728, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f2c7bbfa000

[pid 83988] munmap(0x7f2c7bbfa000, 4218880) = 0

[pid 83988] munmap(0x7f2c80000000, 62889984) = 0

[pid 83988] mprotect(0x7f2c7c000000, 135168, PROT\_READ|PROT\_WRITE) = 0

[pid 83988] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 4

[pid 83988] read(4, "\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

[pid 83988] fstat(4, {st\_mode=S\_IFREG|0644, st\_size=183024, ...}) = 0

[pid 83988] mmap(NULL, 185256, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 4, 0) = 0x7f2c83bcc000

[pid 83988] mmap(0x7f2c83bd0000, 147456, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x4000) = 0x7f2c83bd0000

[pid 83988] mmap(0x7f2c83bf4000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x28000) = 0x7f2c83bf4000

[pid 83988] mmap(0x7f2c83bf8000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 4, 0x2b000) = 0x7f2c83bf8000

[pid 83988] close(4) = 0

[pid 83988] mprotect(0x7f2c83bf8000, 4096, PROT\_READ) = 0

[pid 83988] futex(0x7f2c84e51a58, FUTEX\_WAKE\_PRIVATE, 1) = 1

[pid 83922] <... futex resumed>) = 0

[pid 83988] munmap(0x7f2c84e11000, 20535 <unfinished ...>

[pid 83922] futex(0x7f2c84e51a58, FUTEX\_WAKE\_PRIVATE, 1 <unfinished ...>

[pid 83988] <... munmap resumed>) = 0

[pid 83922] <... futex resumed>) = 0

[pid 83988] futex(0x7f2c83bf9230, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

[pid 83922] rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

[pid 83988] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 83922] <... rt\_sigprocmask resumed>[], 8) = 0

[pid 83988] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 83922] clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f2c843fa990, parent\_tid=0x7f2c843fa990, exit\_signal=0, stack=0x7f2c83bfa000, stack\_size=0x7fff80, tls=0x7f2c843fa6c0} <unfinished ...>

[pid 83988] madvise(0x7f2c843fb000, 8368128, MADV\_DONTNEEDstrace: Process 83989 attached

) = 0

[pid 83989] rseq(0x7f2c843fafe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 83988] exit(0 <unfinished ...>

[pid 83989] <... rseq resumed>) = 0

[pid 83922] <... clone3 resumed> => {parent\_tid=[83989]}, 88) = 83989

[pid 83989] set\_robust\_list(0x7f2c843fa9a0, 24 <unfinished ...>

[pid 83988] <... exit resumed>) = ?

[pid 83989] <... set\_robust\_list resumed>) = 0

[pid 83922] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 83989] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 83988] +++ exited with 0 +++

[pid 83989] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 83922] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 83989] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 83922] mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 83989] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 83922] <... mmap resumed>) = 0x7f2c833cb000

[pid 83989] madvise(0x7f2c83bfa000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 83922] mprotect(0x7f2c833cc000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 83989] <... madvise resumed>) = 0

[pid 83989] exit(0 <unfinished ...>

[pid 83922] <... mprotect resumed>) = 0

[pid 83989] <... exit resumed>) = ?

[pid 83922] rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

[pid 83989] +++ exited with 0 +++

[pid 83922] <... rt\_sigprocmask resumed>[], 8) = 0

[pid 83922] clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f2c83bcb990, parent\_tid=0x7f2c83bcb990, exit\_signal=0, stack=0x7f2c833cb000, stack\_size=0x7fff80, tls=0x7f2c83bcb6c0}strace: Process 83990 attached

<unfinished ...>

[pid 83990] rseq(0x7f2c83bcbfe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 83922] <... clone3 resumed> => {parent\_tid=[83990]}, 88) = 83990

[pid 83990] <... rseq resumed>) = 0

[pid 83922] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 83990] set\_robust\_list(0x7f2c83bcb9a0, 24 <unfinished ...>

[pid 83922] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 83990] <... set\_robust\_list resumed>) = 0

[pid 83922] mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 83990] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 83922] <... mmap resumed>) = 0x7f2c82bca000

[pid 83990] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 83922] mprotect(0x7f2c82bcb000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 83990] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 83922] <... mprotect resumed>) = 0

[pid 83990] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 83922] rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

[pid 83990] madvise(0x7f2c833cb000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 83922] <... rt\_sigprocmask resumed>[], 8) = 0

[pid 83990] <... madvise resumed>) = 0

[pid 83922] clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f2c833ca990, parent\_tid=0x7f2c833ca990, exit\_signal=0, stack=0x7f2c82bca000, stack\_size=0x7fff80, tls=0x7f2c833ca6c0} <unfinished ...>

[pid 83990] exit(0strace: Process 83991 attached

) = ?

[pid 83991] rseq(0x7f2c833cafe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 83990] +++ exited with 0 +++

[pid 83922] <... clone3 resumed> => {parent\_tid=[83991]}, 88) = 83991

[pid 83991] <... rseq resumed>) = 0

[pid 83922] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 83991] set\_robust\_list(0x7f2c833ca9a0, 24 <unfinished ...>

[pid 83922] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 83991] <... set\_robust\_list resumed>) = 0

[pid 83922] futex(0x7f2c833ca990, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 83991, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 83991] rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

[pid 83991] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

[pid 83991] madvise(0x7f2c82bca000, 8368128, MADV\_DONTNEED) = 0

[pid 83991] exit(0) = ?

[pid 83991] +++ exited with 0 +++

[pid 83922] <... futex resumed>) = 0

[pid 83922] write(1, "\nSorted array:\n\0", 16

Sorted array:

) = 16

[pid 83922] write(1, "-90 ", 4-90 ) = 4

[pid 83922] write(1, "-68 ", 4-68 ) = 4

[pid 83922] write(1, "-65 ", 4-65 ) = 4

[pid 83922] write(1, "-60 ", 4-60 ) = 4

[pid 83922] write(1, "-35 ", 4-35 ) = 4

[pid 83922] write(1, "-16 ", 4-16 ) = 4

[pid 83922] write(1, "-4 ", 3-4 ) = 3

[pid 83922] write(1, "14 ", 314 ) = 3

[pid 83922] write(1, "30 ", 330 ) = 3

[pid 83922] write(1, "35 ", 335 ) = 3

[pid 83922] write(1, "37 ", 337 ) = 3

[pid 83922] write(1, "37 ", 337 ) = 3

[pid 83922] write(1, "51 ", 351 ) = 3

[pid 83922] write(1, "62 ", 362 ) = 3

[pid 83922] write(1, "70 ", 370 ) = 3

[pid 83922] write(1, "73 ", 373 ) = 3

[pid 83922] write(1, "74 ", 374 ) = 3

[pid 83922] write(1, "95\n", 395

) = 3

[pid 83922] getpid() = 83922

[pid 83922] fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

[pid 83922] write(1, "You can use 'ps -T -p 83922' to "..., 65You can use 'ps -T -p 83922' to see the threads of this process.

) = 65

[pid 83922] exit\_group(0) = ?

[pid 83922] +++ exited with 0 +++

<... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 83922

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=83922, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

В результате выполнения лабораторной работы удалось познакомиться с многопоточным программированием и реализовать программу, сортирующую массив целых чисел. При выполнении работы возникла проблема с демонстрацией количество потоков, используемого моей программой с помощью стандартных средств операционной системы(вследствии чего не получилось составить таблицу с исследованием работы программы).