Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Группа: М8О-210Б-23

Студент: Кудаева В.А.

Преподаватель: Бахарев В.Д.

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

Дата: 24.12.24

Москва, 2024

**Постановка задачи**

**Вариант 12:**

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, должно быть задано ключом запуска программы. Так же необходимо продемонстрировать количество потоков, используемое программой с помощью стандартных средств операционной системы. Привести исследование зависимости ускорения и эффективности алгоритма от входных данных количества потоков. Объяснить получившиеся результаты.

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

**Общий метод и алгоритм решения**

Использованные системные вызовы:

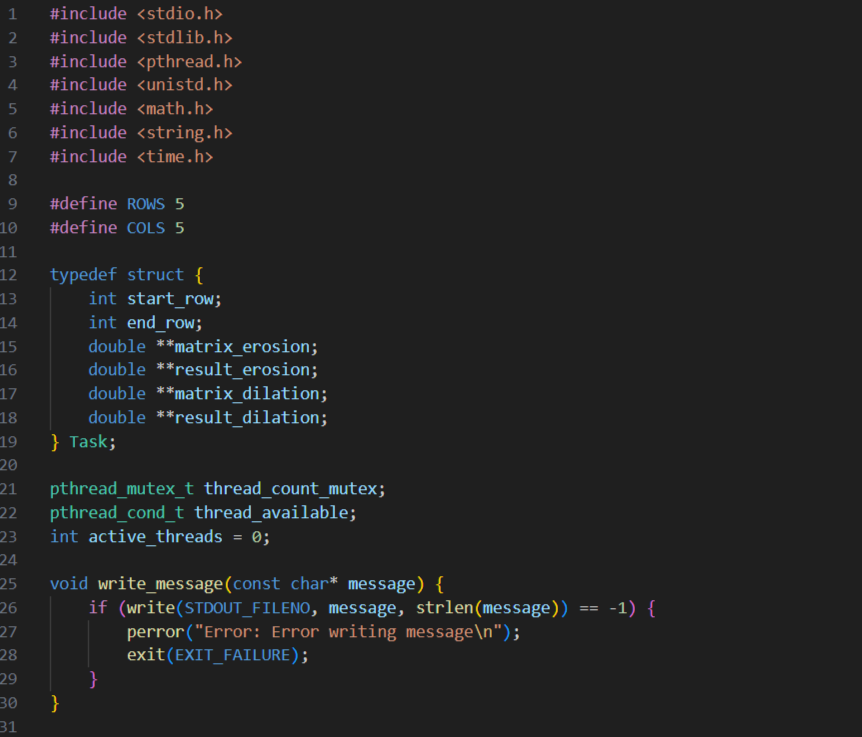
* ssize\_t write(int fd, const void \*buf, size\_t count) – записывает по дескриптору fd count байт из buf;
* int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex) – блокировка мьютекса;
* int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex) – разблокировка мьютекса;
* int pthread\_cond\_wait(pthread\_cond\_t \*cond, pthread\_mutex\_t \*mutex) - переводит поток в ожидающее состояние, пока не будет получен сигнал с помощью условной переменной;
* int pthread\_cond\_signal(pthread\_cond\_t \*cond) - отправляет сигнал с помощью условной переменной;
* int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine) (void \*), void \*arg) – создание потока;
* int pthread\_join(pthread\_t thread, void \*\*retval) – ожидание завершения потока;
* void exit(int status) – завершение программы с заданным кодом возврата.

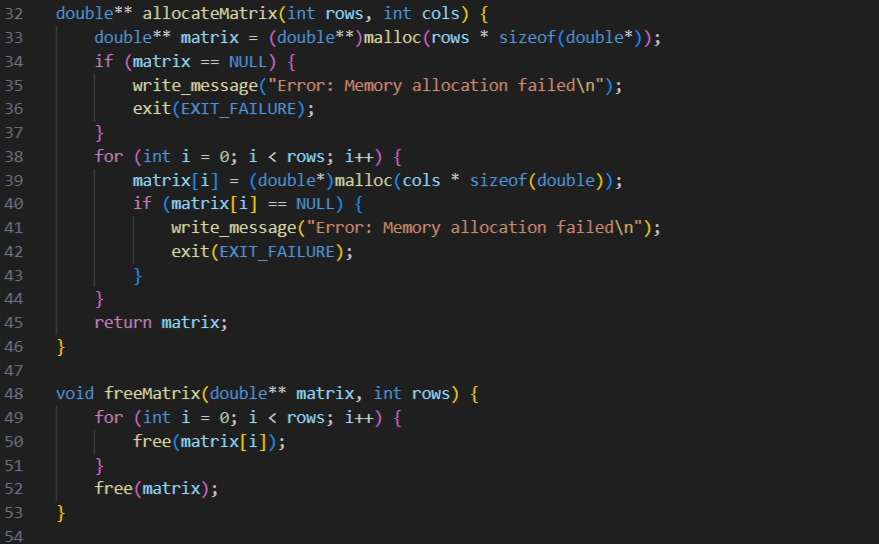
Программа предназначена для выполнения операций эрозии и наращивания над матрицей заданное количество раз. Она работает в многопоточном режиме, используя мьютексы и условные переменные для синхронизации потоков.

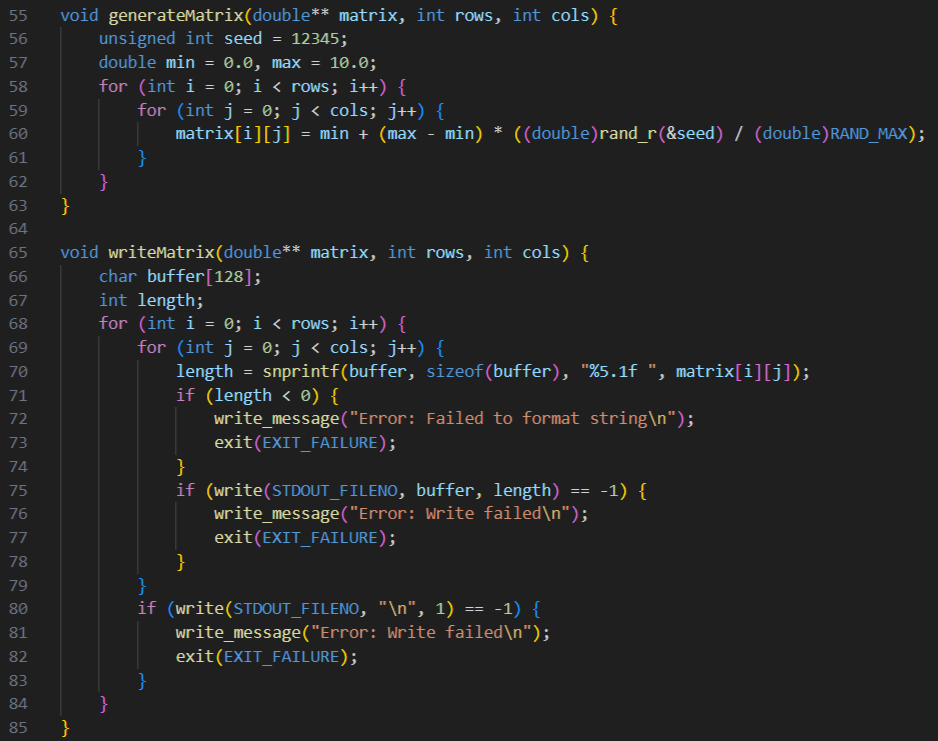
В начале каждой итерации вычисляется количество потоков, которые могут быть задействованы. Если число потоков больше количества строк в матрице, оно уменьшается до количества строк, чтобы избежать выделения потоков, которые не будут выполнять полезной работы. Массивы threads и tasks создаются для хранения идентификаторов потоков и данных, которые будут переданы каждому потоку. Затем вычисляется количество строк, которое будет обрабатываться каждым потоком.

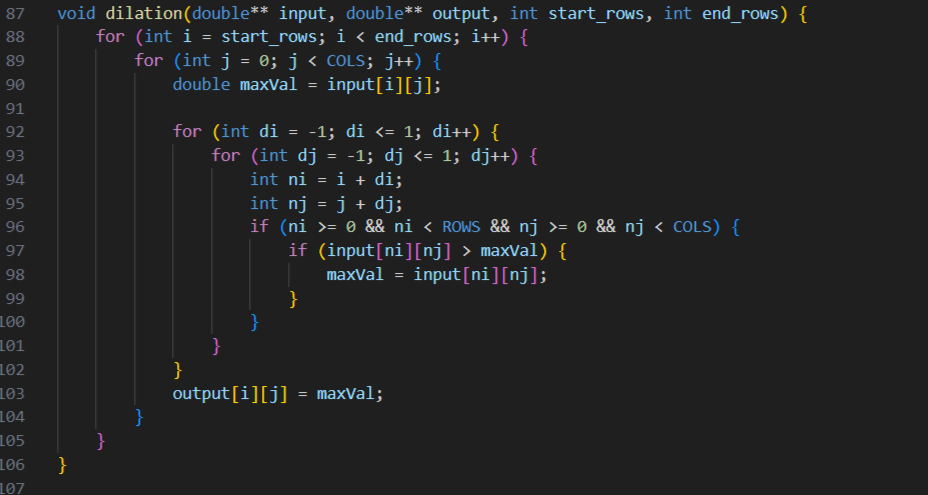
Цикл распределяет строки между потоками. Сначала поток ожидает своей очереди, если текущее количество активных потоков достигло лимита. Мьютекс thread\_count\_mutex блокируется для защиты глобальной переменной active\_threads, которая отслеживает **количество потоков**, выполняющихся в данный момент времени. Она используется для ограничения числа одновременно работающих потоков до значения max\_threads. Поток может продолжить выполнение только после получения сигнала об освобождении потока через условную переменную thread\_available. Далее заполняется структура Task для текущей итерации. Функция process\_task выполняет операцию эрозии и наращивания на указанном диапазоне строк. После запуска всех потоков текущей итерации программа ожидает их завершения. Функция pthread\_join блокирует выполнение основного потока до тех пор, пока соответствующий поток не завершит работу. После завершения всех потоков результаты операций эрозии и наращивания копируются обратно в исходные матрицы.

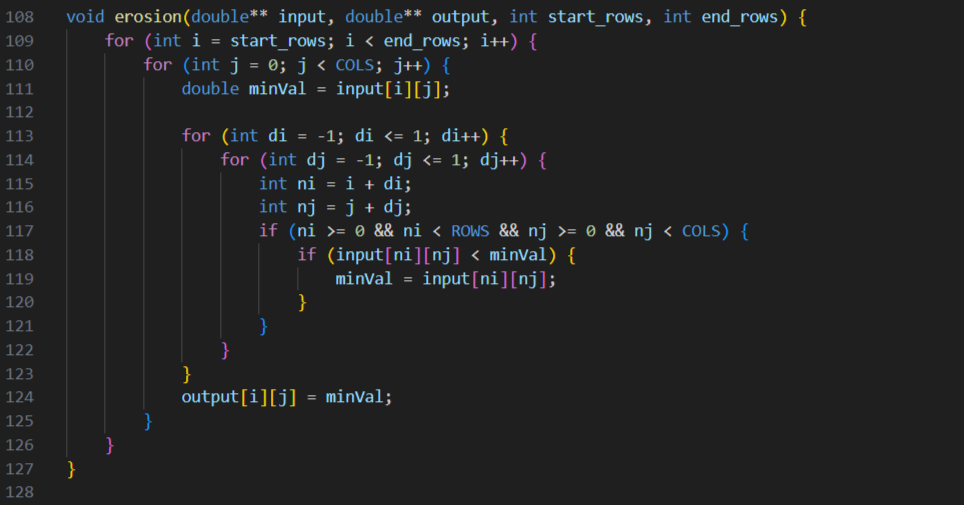
**Код программы**

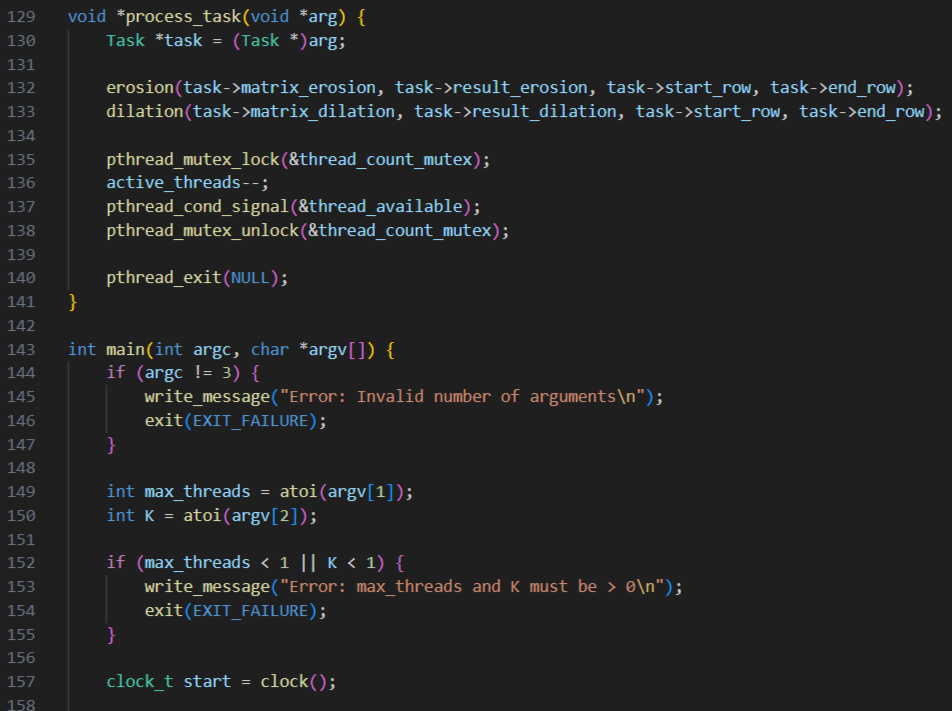
****

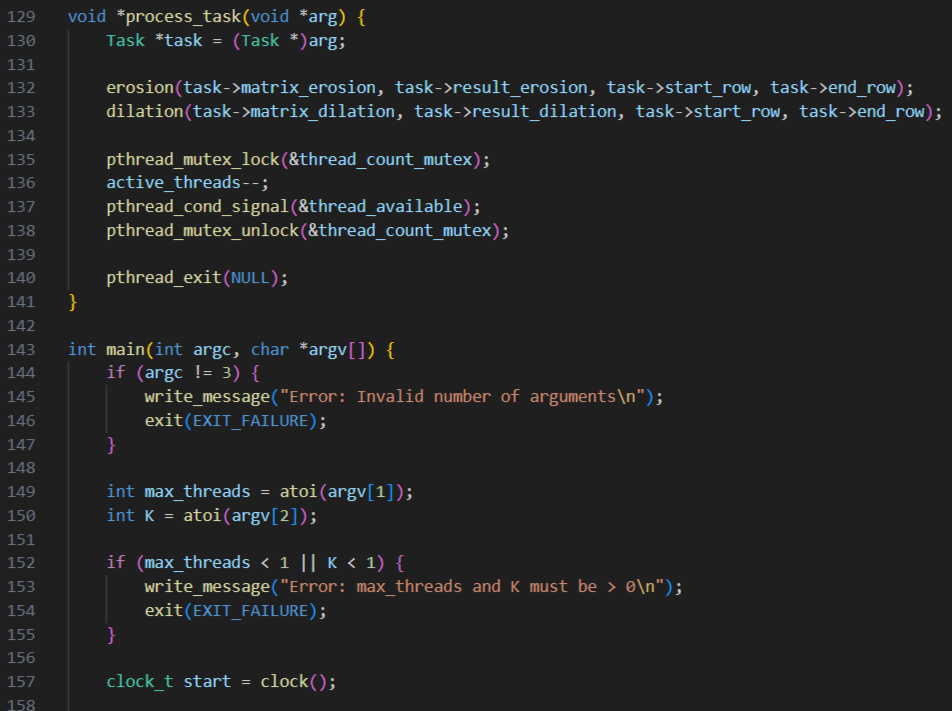
****

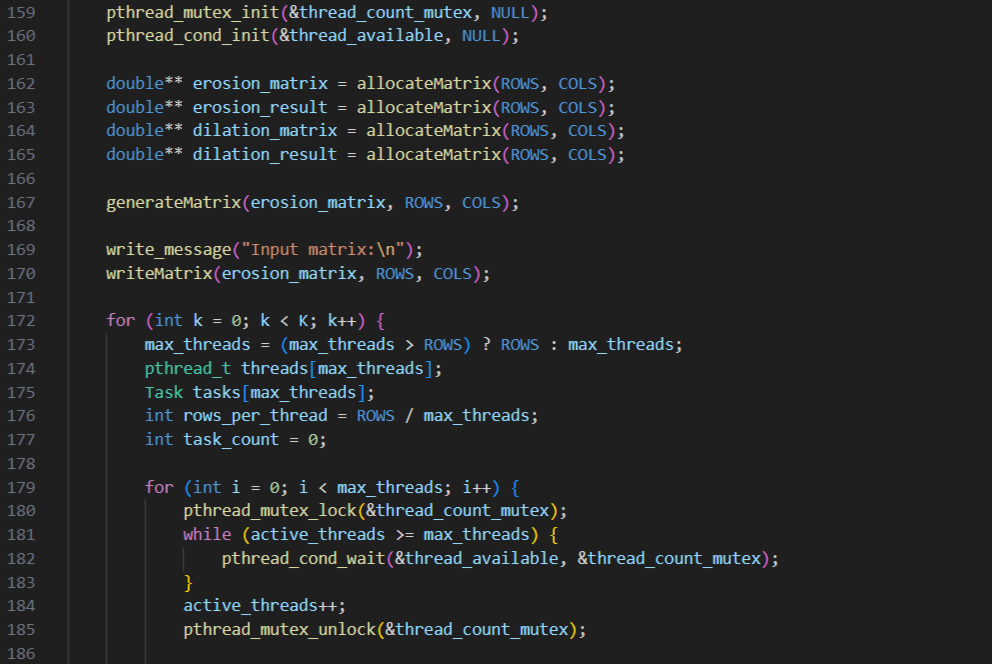
****

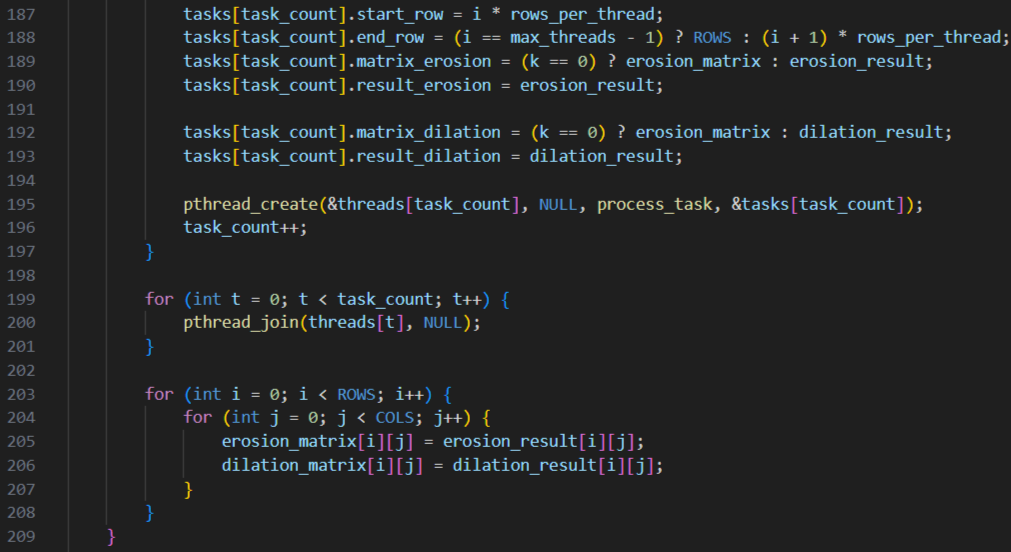
****

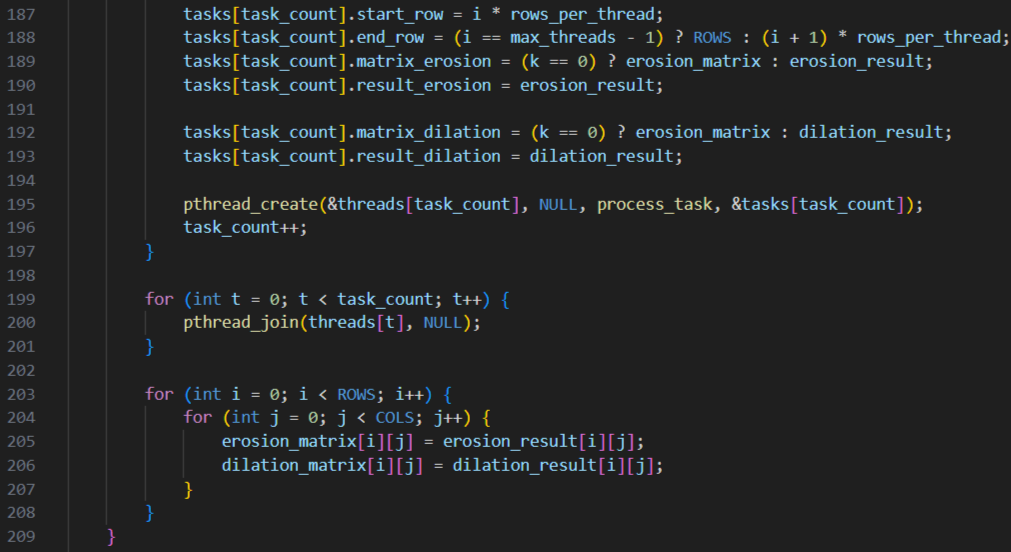
****

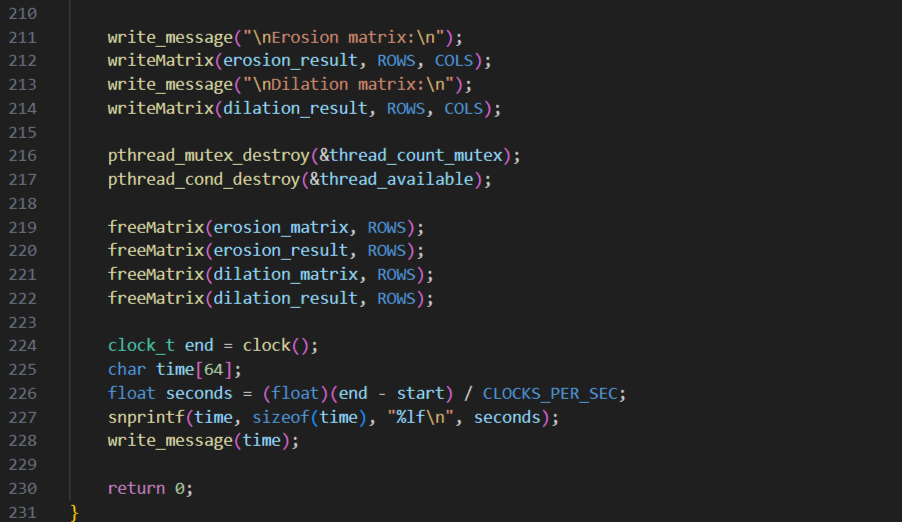
****

****

****

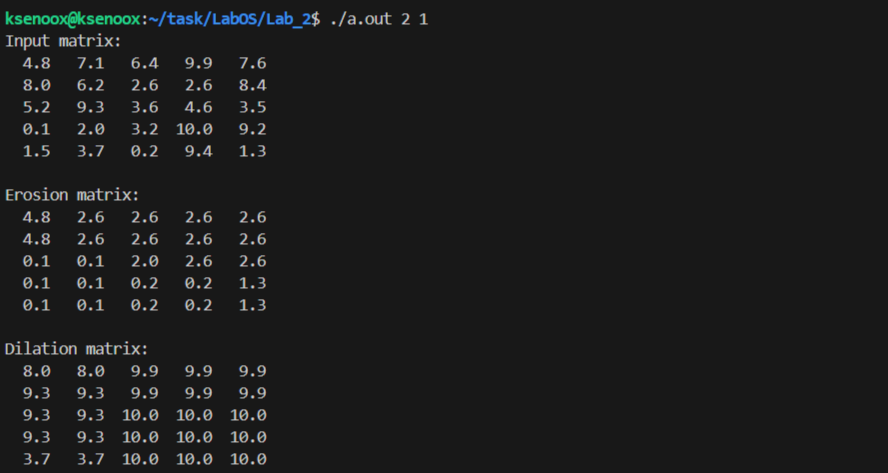
****

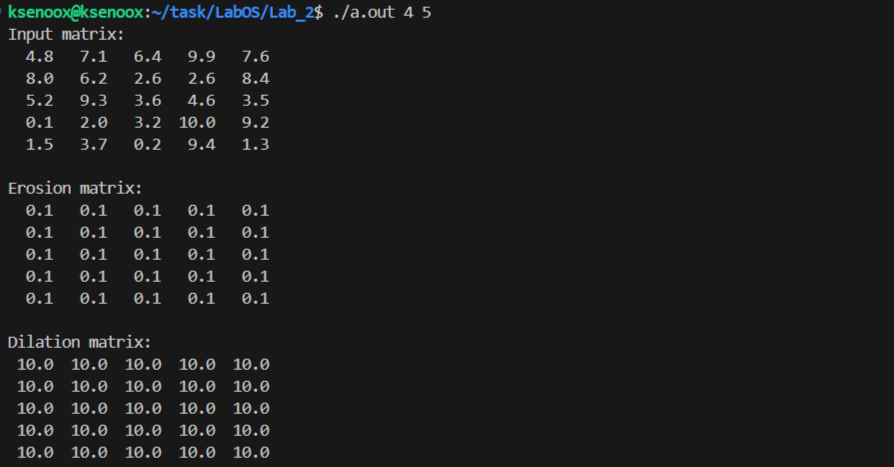
****

****

**Протокол работы программы**

**Тестирование**

****

****

|  |  |  |  |
| --- | --- | --- | --- |
| Число потоков | Время выполнения, мс | Ускорение | Эффективность |
| 1 | 823 | 1 | 1 |
| 2 | 688 | 1, 19 | 0.59 |
| 3 | 623 | 1,32 | 0.44 |
| 4 | 579 | 1,42 | 0.35 |
| 5 | 511 | 1,61 | 0.32 |

**Объяснение**

Многопоточность позволяет ускорить выполнение программы, так как задачи разделяются между несколькими потоками и выполняются параллельно. При увеличении числа потоков растут накладные расходы на управление потоками, например, создание, синхронизация, переключение контекста, поэтому эффективность снижается. Для **максимальной производительности** количество потоков должно быть равно количеству **физических ядер.** Когда количество потоков **превышает количество ядер процессора,** потоки перестают выполняться **физически параллельно**. Вместо этого процессор вынужден **переключать контекст** между потоками, что способствует снижению эффективности.

**Вывод strace**

execve("./a.out", ["./a.out", "4", "1"], 0x7ffebc72c090 /\* 74 vars \*/) = 0

brk(NULL) = 0x55f03d310000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffe601c7030) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f6e97335000

newfstatat(AT\_FDCWD, "/usr/lib/openfoam/openfoam2312/platforms/linux64GccDPInt32Opt/lib/dummy", {st\_mode=S\_IFDIR|0755, st\_size=4096, ...}, 0) = 0

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=27331, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 27331, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f6e9732e000

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\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 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) = 0x7f6e97105000

mprotect(0x7f6e9712d000, 2023424, PROT\_NONE) = 0

mmap(0x7f6e9712d000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f6e9712d000

mmap(0x7f6e972c2000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f6e972c2000

mmap(0x7f6e9731b000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7f6e9731b000

mmap(0x7f6e97321000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f6e97321000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f6e97102000

arch\_prctl(ARCH\_SET\_FS, 0x7f6e97102740) = 0

set\_tid\_address(0x7f6e97102a10) = 357885

set\_robust\_list(0x7f6e97102a20, 24) = 0

rseq(0x7f6e971030e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f6e9731b000, 16384, PROT\_READ) = 0

mprotect(0x55f01ae3a000, 4096, PROT\_READ) = 0

mprotect(0x7f6e9736f000, 8192, PROT\_READ) = 0

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

munmap(0x7f6e9732e000, 27331) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=16167500}) = 0

getrandom("\x1f\xbb\x0f\x44\x86\x65\x4c\x6c", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55f03d310000

brk(0x55f03d331000) = 0x55f03d331000

write(1, "Input matrix:\n", 14Input matrix:

) = 14

write(1, " 4.8 ", 6 4.8 ) = 6

write(1, " 7.1 ", 6 7.1 ) = 6

write(1, " 6.4 ", 6 6.4 ) = 6

write(1, " 9.9 ", 6 9.9 ) = 6

write(1, " 7.6 ", 6 7.6 ) = 6

write(1, "\n", 1

) = 1

write(1, " 8.0 ", 6 8.0 ) = 6

write(1, " 6.2 ", 6 6.2 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, " 8.4 ", 6 8.4 ) = 6

write(1, "\n", 1

) = 1

write(1, " 5.2 ", 6 5.2 ) = 6

write(1, " 9.3 ", 6 9.3 ) = 6

write(1, " 3.6 ", 6 3.6 ) = 6

write(1, " 4.6 ", 6 4.6 ) = 6

write(1, " 3.5 ", 6 3.5 ) = 6

write(1, "\n", 1

) = 1

write(1, " 0.1 ", 6 0.1 ) = 6

write(1, " 2.0 ", 6 2.0 ) = 6

write(1, " 3.2 ", 6 3.2 ) = 6

write(1, " 10.0 ", 6 10.0 ) = 6

write(1, " 9.2 ", 6 9.2 ) = 6

write(1, "\n", 1

) = 1

write(1, " 1.5 ", 6 1.5 ) = 6

write(1, " 3.7 ", 6 3.7 ) = 6

write(1, " 0.2 ", 6 0.2 ) = 6

write(1, " 9.4 ", 6 9.4 ) = 6

write(1, " 1.3 ", 6 1.3 ) = 6

write(1, "\n", 1

) = 1

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f6e97196870, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f6e97147520}, NULL, 8) = 0

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

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f6e96901000

mprotect(0x7f6e96902000, 8388608, PROT\_READ|PROT\_WRITE) = 0

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

clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f6e97101910, parent\_tid=0x7f6e97101910, exit\_signal=0, stack=0x7f6e96901000, stack\_size=0x7fff00, tls=0x7f6e97101640} => {parent\_tid=[357886]}, 88) = 357886

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

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f6e96100000

mprotect(0x7f6e96101000, 8388608, PROT\_READ|PROT\_WRITE) = 0

futex(0x7f6e97371a98, FUTEX\_WAIT\_PRIVATE, 2, NULL) = 0

futex(0x7f6e97371a98, FUTEX\_WAKE\_PRIVATE, 1) = 0

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

clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f6e96900910, parent\_tid=0x7f6e96900910, exit\_signal=0, stack=0x7f6e96100000, stack\_size=0x7fff00, tls=0x7f6e96900640} => {parent\_tid=[357887]}, 88) = 357887

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

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f6e958df000

mprotect(0x7f6e958e0000, 8388608, PROT\_READ|PROT\_WRITE) = 0

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

clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f6e960df910, parent\_tid=0x7f6e960df910, exit\_signal=0, stack=0x7f6e958df000, stack\_size=0x7fff00, tls=0x7f6e960df640} => {parent\_tid=[0]}, 88) = 357888

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

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f6e950de000

mprotect(0x7f6e950df000, 8388608, PROT\_READ|PROT\_WRITE) = 0

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

clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f6e958de910, parent\_tid=0x7f6e958de910, exit\_signal=0, stack=0x7f6e950de000, stack\_size=0x7fff00, tls=0x7f6e958de640} => {parent\_tid=[357889]}, 88) = 357889

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

write(1, "\nErosion matrix:\n", 17

Erosion matrix:

) = 17

write(1, " 4.8 ", 6 4.8 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, "\n", 1

) = 1

write(1, " 4.8 ", 6 4.8 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, "\n", 1

) = 1

write(1, " 0.1 ", 6 0.1 ) = 6

write(1, " 0.1 ", 6 0.1 ) = 6

write(1, " 2.0 ", 6 2.0 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, " 2.6 ", 6 2.6 ) = 6

write(1, "\n", 1

) = 1

write(1, " 0.1 ", 6 0.1 ) = 6

write(1, " 0.1 ", 6 0.1 ) = 6

write(1, " 0.2 ", 6 0.2 ) = 6

write(1, " 0.2 ", 6 0.2 ) = 6

write(1, " 1.3 ", 6 1.3 ) = 6

write(1, "\n", 1

) = 1

write(1, " 0.1 ", 6 0.1 ) = 6

write(1, " 0.1 ", 6 0.1 ) = 6

write(1, " 0.2 ", 6 0.2 ) = 6

write(1, " 0.2 ", 6 0.2 ) = 6

write(1, " 1.3 ", 6 1.3 ) = 6

write(1, "\n", 1

) = 1

write(1, "\nDilation matrix:\n", 18

Dilation matrix:

) = 18

write(1, " 8.0 ", 6 8.0 ) = 6

write(1, " 8.0 ", 6 8.0 ) = 6

write(1, " 9.9 ", 6 9.9 ) = 6

write(1, " 9.9 ", 6 9.9 ) = 6

write(1, " 9.9 ", 6 9.9 ) = 6

write(1, "\n", 1

) = 1

write(1, " 9.3 ", 6 9.3 ) = 6

write(1, " 9.3 ", 6 9.3 ) = 6

write(1, " 9.9 ", 6 9.9 ) = 6

write(1, " 9.9 ", 6 9.9 ) = 6

write(1, " 9.9 ", 6 9.9 ) = 6

write(1, "\n", 1

) = 1

write(1, " 9.3 ", 6 9.3 ) = 6

write(1, " 9.3 ", 6 9.3 ) = 6

write(1, " 10.0 ", 6 10.0 ) = 6

write(1, " 10.0 ", 6 10.0 ) = 6

write(1, " 10.0 ", 6 10.0 ) = 6

write(1, "\n", 1

) = 1

write(1, " 9.3 ", 6 9.3 ) = 6

write(1, " 9.3 ", 6 9.3 ) = 6

write(1, " 10.0 ", 6 10.0 ) = 6

write(1, " 10.0 ", 6 10.0 ) = 6

write(1, " 10.0 ", 6 10.0 ) = 6

write(1, "\n", 1

) = 1

write(1, " 3.7 ", 6 3.7 ) = 6

write(1, " 3.7 ", 6 3.7 ) = 6

write(1, " 10.0 ", 6 10.0 ) = 6

write(1, " 10.0 ", 6 10.0 ) = 6

write(1, " 10.0 ", 6 10.0 ) = 6

write(1, "\n", 1

) = 1

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=0, tv\_nsec=24735000}) = 0

write(1, "0.008568\n", 90.008568

) = 9

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

В ходе выполнения лабораторной работы мне удалось написать многопоточную программу для выполнения операций **эрозии** и **наращивания** матрицы с использованием синхронизации потоков. Программа была протестирована на различных количествах потоков, и получены экспериментальные данные о времени выполнения, ускорении и эффективности.