**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: Абросимов Алексей Дмитриевич

Группа: М8О-207Б-20

Преподаватель: Е. С. Миронов

Вариант: 5

Дата:

Оценка:

Москва, 2021

## **1. Описание**

Данная лабораторная работа будет выполняться в ОС Unix.

**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

**Использованные системные вызовы:**

**pid\_t pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine) (void), void \*arg); -** Данная функция запускает новый поток в вызывающем процессе. Новый поток начинает исполнение, вызывая start\_routine(void \*arg).

**pid\_t pthread\_join(pthread\_t thread, void \*\*retval);-** Данная функция ожидает завершения указанного потока. Если этот поток уже завершен, то данная функция немедленно возвращается

Tp – время выполнения на p различных потоках/вычислительных ядрах

Sp – ускорение. Sp = T1/Tn, Sp < p

Xp – эффективность/загруженность. Xp = Sp/p, Xp<1

Заполняем таблицу для сортировки вектора из 10000 элементов.

|  |  |  |  |
| --- | --- | --- | --- |
| Num\_of\_threads | Tn(sec) | Ускорение | Эффективность |
| 1 | 0.808 | 1 | 1 |
| 2 | 0.666 | 1,2132 | 0.6 |
| 4 | 0.675 | 1,19 | 0.29 |
| 10 | 0.904 | 0.89 | 0.08 |

## **2. Исходный код**

#include <iostream>

#include <pthread.h>

#include <vector>

#include <assert.h>

#include <chrono>

void sort\_betcher(std::vector<int> &arr, int maxth, int res);

void \*sort\_thread(void \*init);

void sort(std::vector<int> &arr, int num);

int compare(int &a, int &b)

{

if (a > b)

{

std::swap(a, b);

return 1;

}

return 0;

}

int chet(std::vector<int> &arr,int deep)

{

int result = 0;

for (unsigned int i = 2\*(deep%2); i < arr.size() - 1; i += 4)

{

result += compare(arr[i], arr[i + 1]);

}

if (result > 0)

return 1;

else

return 0;

}

int nechet(std::vector<int> &arr,int deep)

{

int result = 0;

for (unsigned int i = 1+2\*(deep%2); i < arr.size() - 1; i += 4)

{

result += compare(arr[i], arr[i + 1]);

}

if (result > 0)

return 1;

else

return 0;

}

struct sort\_starter

{

std::vector<int> &arr;

int deep;

int result;

};

void \*sort\_thread(void \*init)

{

sort\_starter \*start = (sort\_starter \*)init;

sort\_betcher(start->arr, start->deep, start->result);

return NULL;

}

void sort\_betcher(std::vector<int> &arr, int deep, int res)

{

if (res)

{

res = chet(arr,deep) + nechet(arr,deep);

if (deep-- >0)

{

pthread\_t thread;

sort\_starter arg = {arr, deep, res};

int ret = pthread\_create(&thread, NULL, sort\_thread, &arg);

assert(ret == 0);

sort\_betcher(arr, deep, res);

pthread\_join(thread, NULL);

}

else

sort(arr, 1);

}

}

void sort(std::vector<int> &arr, int res)

{

res = chet(arr,0) + nechet(arr,0) + chet(arr,1) + nechet(arr,1);

if (res)

sort(arr, res);

}

int main()

{

int maxth;

while(true){

std::cout << "Insert max num of threads: ";

std::cin >> maxth;

if(maxth>0) break;

else printf("Maxnum must be more than 0\n");

}

std::vector<int> arr;

int count;

while(true){

std::cout << "Insert length of vector: ";

std::cin >> count;

if(count>0) break;

else printf("Length must be more than 0\n");

}

srand(17327);

if (count == 2)

{

for (int i = 0; i < count; i++)

arr.push\_back(rand() % 1000);

compare(arr[0], arr[1]);

}

else

{

for (int i = 0; i < count; i++)

arr.push\_back(rand() % 1000);

}

for (unsigned int i = 0; i < arr.size(); i++)

printf("%d ", arr[i]);

printf("\n");

printf("\n");

auto first\_time = std::chrono::high\_resolution\_clock::now();

sort\_betcher(arr, maxth, 1);

for (unsigned int i = 0; i < arr.size(); i++)

printf("%d ", arr[i]);

printf("\n");

auto last\_time = std::chrono::high\_resolution\_clock::now();

auto myperiod = std::chrono::duration\_cast<std::chrono::milliseconds>(last\_time - first\_time).count();

std::cout << "Time: " << myperiod << " milliseconds" << std::endl;

return 0;

}

## **3. Консоль**

**$ strace ./prog**

execve("./prog", ["./prog"], 0x7ffc17de3ea0 /\* 61 vars \*/) = 0

brk(NULL) = 0x559d602cb000

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

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=104259, ...}) = 0

mmap(NULL, 104259, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fe27f9de000

close(3) = 0

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

openat(AT\_FDCWD, "/usr/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\220\304\10\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1594864, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe27f9dc000

mmap(NULL, 3702848, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fe27f446000

mprotect(0x7fe27f5bf000, 2097152, PROT\_NONE) = 0

mmap(0x7fe27f7bf000, 49152, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x179000) = 0x7fe27f7bf000

mmap(0x7fe27f7cb000, 12352, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fe27f7cb000

close(3) = 0

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

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\300\*\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=96616, ...}) = 0

mmap(NULL, 2192432, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fe27f22e000

mprotect(0x7fe27f245000, 2093056, PROT\_NONE) = 0

mmap(0x7fe27f444000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x16000) = 0x7fe27f444000

close(3) = 0

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

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpthread.so.0", 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\0000b\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=144976, ...}) = 0

mmap(NULL, 2221184, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fe27f00f000

mprotect(0x7fe27f029000, 2093056, PROT\_NONE) = 0

mmap(0x7fe27f228000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19000) = 0x7fe27f228000

mmap(0x7fe27f22a000, 13440, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fe27f22a000

close(3) = 0

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

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\20\35\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2030928, ...}) = 0

mmap(NULL, 4131552, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fe27ec1e000

mprotect(0x7fe27ee05000, 2097152, PROT\_NONE) = 0

mmap(0x7fe27f005000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7fe27f005000

mmap(0x7fe27f00b000, 15072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fe27f00b000

close(3) = 0

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

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\200\272\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1700792, ...}) = 0

mmap(NULL, 3789144, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fe27e880000

mprotect(0x7fe27ea1d000, 2093056, PROT\_NONE) = 0

mmap(0x7fe27ec1c000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19c000) = 0x7fe27ec1c000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe27f9da000

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fe27f9d7000

arch\_prctl(ARCH\_SET\_FS, 0x7fe27f9d7740) = 0

mprotect(0x7fe27f005000, 16384, PROT\_READ) = 0

mprotect(0x7fe27ec1c000, 4096, PROT\_READ) = 0

mprotect(0x7fe27f228000, 4096, PROT\_READ) = 0

mprotect(0x7fe27f444000, 4096, PROT\_READ) = 0

mprotect(0x7fe27f7bf000, 40960, PROT\_READ) = 0

mprotect(0x559d5fd65000, 4096, PROT\_READ) = 0

mprotect(0x7fe27f9f8000, 4096, PROT\_READ) = 0

munmap(0x7fe27f9de000, 104259) = 0

set\_tid\_address(0x7fe27f9d7a10) = 3419

set\_robust\_list(0x7fe27f9d7a20, 24) = 0

rt\_sigaction(SIGRTMIN, {sa\_handler=0x7fe27f014cb0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7fe27f021980}, NULL, 8) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7fe27f014d50, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7fe27f021980}, NULL, 8) = 0

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

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

brk(NULL) = 0x559d602cb000

brk(0x559d602ec000) = 0x559d602ec000

futex(0x7fe27f7cc09c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

futex(0x7fe27f7cc0a8, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 2), ...}) = 0

write(1, "Insert length of vector: ", 25Insert length of vector: ) = 25

fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 2), ...}) = 0

read(0, 10

"10\n", 1024) = 3

write(1, "164 745 653 116 167 617 473 928 "..., 40164 745 653 116 167 617 473 928 96 304

) = 40

write(1, "\n", 1

) = 1

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7fe27e07f000

mprotect(0x7fe27e080000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7fe27e87efb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7fe27e87f9d0, tls=0x7fe27e87f700, child\_tidptr=0x7fe27e87f9d0) = 3431

clone(child\_stack=0x7fe27e07dfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7fe27e07e9d0, tls=0x7fe27e07e700, child\_tidptr=0x7fe27e07e9d0) = 3433

futex(0x7fe27e07e9d0, FUTEX\_WAIT, 3433, NULL) = 0

write(1, "96 116 164 167 304 473 617 653 7"..., 4096 116 164 167 304 473 617 653 745 928

) = 40

write(1, "Time: 0 milliseconds\n", 21Time: 0 milliseconds

) = 21

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

exit\_group(0) = ?

+++ exited with 0 +++

**$ ./prog**

Insert length of vector: 10

164 745 653 116 167 617 473 928 96 304

96 116 164 167 304 473 617 653 745 928

Time: 0 milliseconds

**4. Вывод**

Данная лабораторная работа позволила мне ознакомиться с методами работы с многопоточностью, что невероятно полезно при написании современных программ, тк позволяет ускорить время выполнения иногда в десятки раз. Стоит отметить, что при создании поток имеет доступ ко всему адресному пространству процесса, в котором он создан, в том числе к таблице дескрипторов, к отправляемым процессу сигналам и, что важно, к данным других потоков. Разные потоки могут производить чтение/запись в одинаковые блоки памяти, что, с одной стороны улучшает производительность параллельных вычислений(можно создать один большой блок памяти, где потоки будут обращаться только к своей части данных), а с другой стороны, может произойти race condition, при котором один поток может считать данные, затем в них после чтения сразу запишет данные другой поток, затем первый снова запишет данные, но на основании тех, что он считал. Следовательно, данные, записанные вторым потоком, потеряются.