Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу

«Операционные системы»

Группа: М80 – 201Б-19
Студент: Цыкин И.А.
Преподаватель: Миронов Е.С.
Оценка:

_____

_____

Дата:

_____

Москва, 2020.

Содержание

1 Постановка задачи
2 Общие сведения о программе
3 Общий метод и алгоритм решения
4 Листинг программы
5 Результаты работы программы
6 Strace
7 Вывод

Постановка задачи

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Uni. 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.

15 вариант: Перемножение полиномов. На вход подается N-полиномов, необходимо их перемножить

Общие сведения о программе

Программа компилируется из одного файла threads.c
Для работы программы необходимо использование библиотек pthread.h и time.h
Библиотека pthread.h - библиотечка стандарта POSIX, которая организует потоки внутри процесса, time.h — библиотека, содержащая типы и функции для работы с датой и временем.
Основные функции:

1. **int pthread_mutex_init(pthread_mutex_t *mutex, constpthread_mutexattr_t *attr);** - инициализация мьютекса
2. **int pthread_mutex_destroy(pthread_mutex_t *mutex);** - удалениемьютекса
3. **int pthread_mutex_lock(pthread_mutex_t *mutex);** - захват функции, когда затрагивается критическая область в памяти
4. **int pthread_mutex_unlock(pthread_mutex_t *mutex);** - освобождение
5. **int pthread_create (pthread_t * thread, const pthread_attr_t * attr, void * (*start_routine)(void *), void *arg);** - создания нового потока
6. **int pthread_join(pthread_t th, void **thread_return);** - ожиданиезавершения процеса
7. **clock();** - функция захватат времени

Общий метод и алгоритм решения.

В данной задаче я записываю полином в виде массива чисел определенного размера. Если один полином состоит из n элементов, а второй — из m, то результирующий полином будет иметь n+m-1 элементов. Тогда можно создать 3 массива один состоит из n, второй — из m, а третий — из n+m-1 элементов. При данном количестве потоков(пусть будет) можно разбить число элементов n на приблизительно равные части. Тогда каждый поток будет перемножать коэфициенты и заполнять результирующий массив, но при этом будет проблема с областью памяти, тогда необходимо будет блокировать процессы, когда будет затронута кртитическая область памяти.

Листинг программы

parent.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <time.h>

#define THREAD_NUMBER_ERROR -1
#define THREAD_CREATE_ERROR -2
#define THREAD_JOIN_ERROR -3

pthread_mutex_t mutex;

typedef struct{
	int* first;
	int* second;
	int* result;
	int n, m, lhs, rhs;
}Data;

void* thread_func(void* arg){
	Data* data = (Data*) arg;
	int result[data->n + data->m - 1];
	for (int i = data->lhs; i < data->rhs && i < data->n; ++i) {
		for (int j = 0; j < data->m; ++j) {
			result[i + j] += data->first[i] * data->second[j];
		}
	}

	pthread_mutex_lock(&mutex);
	for(int i = 0; i < data->n+data->m - 1; ++i){
		data->result[i] += result[i];
	}
	pthread_mutex_unlock(&mutex);

	return NULL;
}

int main(int argc, char* argv[]){
	if (argc<2) {
		printf("You must use ./... number of threads \n");
		return 0;
	}
	int thread_number = atoi(argv[1]);
	int n, m;
	printf("Thread number: %d\n", thread_number);
	printf("Enter the number of degree of 1 polynomial: ");
	scanf("%d", &n);
```

```
int* first = malloc(sizeof(int) * n);
for(int i = n-1; i >= 0; --i){
      scanf("%d", &first[i]);
}
printf("Enter the number of degree of 2 polynomial: ");
scanf("%d", &m);
int* second = malloc(sizeof(int) * m);
for(int i = m-1; i >= 0; --i){
      scanf("%d", &second[i]);
}
int* result = malloc(sizeof(int) * (n+m-1));
for(int i = 0; i < n+m; ++i){
      result[i] = 0;
}
pthread_t* thread = (pthread_t*)malloc(sizeof(pthread_t) *
thread_number);
pthread_mutex_init(&mutex, NULL);
Data* data = (Data*)malloc(sizeof(Data) * thread_number);
int k = (n + thread_number - 1) / thread_number;
for (int i = 0; i < thread_number; ++i) {
      data[i].first = first;
      data[i].second = second;
      data[i].result = result;
      data[i].lhs = i * k;
      data[i].rhs = (i + 1) * k;
      data[i].n = n;
      data[i].m = m;
}
double t0 = clock();
for (int i = 0; i < thread_number; ++i) {
      if (pthread_create(&thread[i], NULL, thread_func, &data[i])) {
            printf("Error creating thread!\n");
            return THREAD_CREATE_ERROR;
      }
}
for (int i = 0; i < thread_number; ++i) {
      if (pthread_join(thread[i], NULL)) {
            printf("Error executing thread!\n");
            return THREAD_JOIN_ERROR;
      }
}
double t1 = clock();
printf("Result: ");
for (int i = n + m - 2; i >= 0; --i) {
      printf("%d ", result[i]);
}
printf("\n");
printf("Execution time %lf ms\n", (t1 - t0) / 1000.0);
free(first);
free(second);
free(result);
```

```
        free(thread)
        ;free(data);
        pthread_mutex_destroy(&mutex);
        return 0;
}
```

Результаты работы программы

```
vaney@vaney-VirtualBox:~$ cd */lab3
vaney@vaney-VirtualBox:~/OS/lab3$ ./main 1
Thread number: 1
Enter the number of degree of 1 polynomial: 6
1 2 4 8 16 32
Enter the number of degree of 2 polynomial: 3
1 1 1
Result: 1 3 7 14 28 56 48 32
Execution time 0.144000 ms
vaney@vaney-VirtualBox:~/OS/lab3$ ./main 2
Thread number: 2
Enter the number of degree of 1 polynomial: 6
1 2 4 8 16 32
Enter the number of degree of 2 polynomial: 3
1 1 1
Result: 1 3 7 14 28 56 48 32
Execution time 0.096000 ms
vaney@vaney-VirtualBox:~/OS/lab3$ ./main 3
Thread number: 3
Enter the number of degree of 1 polynomial: 6
1 2 4 8 16 32
Enter the number of degree of 2 polynomial: 3
1 1 1
Result: 1 3 7 14 28 56 48 32
Execution time 0.127000 ms
vaney@vaney-VirtualBox:~/OS/lab3$ ./main 6
Thread number: 6
Enter the number of degree of 1 polynomial: 6
1 2 4 8 16 32
Enter the number of degree of 2 polynomial: 3
1 1 1
Result: 1 3 7 14 28 56 48 32
Execution time 0.212000 ms
vaney@vaney-VirtualBox:~/OS/lab3$ ./main 1000
Thread number: 1000
Enter the number of degree of 1 polynomial: 6
1 2 4 8 16 32
Enter the number of degree of 2 polynomial: 3
1 1 1
Result: 1 3 7 14 28 56 48 32
Execution time 33.006000 ms
vaney@vaney-VirtualBox:~/OS/lab3$ ./main 1
Thread number: 1
Enter the number of degree of 1 polynomial: 8
1 2 3 4 5 6 7 8
Enter the number of degree of 2 polynomial: 5
1 1 1 1 1
Result: 1 3 6 10 15 20 25 30 26 21 15 8
Execution time 0.107000 ms
vaney@vaney-VirtualBox:~/OS/lab3$ ./main 2
Thread number: 2
Enter the number of degree of 1 polynomial: 8
1 2 3 4 5 6 7 8
Enter the number of degree of 2 polynomial: 5
1 1 1 1 1
Result: 1 3 6 10 15 20 25 30 26 21 15 8
Execution time 0.096000 ms
```

```
vaney@vaney-VirtualBox:~/OS/lab3$ ./main 4
Thread number: 4
Enter the number of degree of 1 polynomial: 8
1 2 3 4 5 6 7 8
Enter the number of degree of 2 polynomial: 5
1 1 1 1 1
Result: 1 3 6 10 15 20 25 30 26 21 15 8
Execution time 0.214000 ms
vaney@vaney-VirtualBox:~/OS/lab3$ ./main 8
Thread number: 8
Enter the number of degree of 1 polynomial: 8
1 2 3 4 5 6 7 8
Enter the number of degree of 2 polynomial: 5
1 1 1 1 1
Result: 1 3 6 10 15 20 25 30 26 21 15 8
Execution time 0.492000 ms
vaney@vaney-VirtualBox:~/OS/lab3$ ./main 16
Thread number: 16
Enter the number of degree of 1 polynomial: 8
1 2 3 4 5 6 7 8
Enter the number of degree of 2 polynomial: 5
1 1 1 1 1
Result: 1 3 6 10 15 20 25 30 26 21 15 8
Execution time 0.698000 ms
```

Сравнение результатов работы

Для сравнения работы разного количества потоков я перемножал 2 полинома 36 степени. Мной была составлена таблица времени, ускорения и эффективности.

N	Время	Ускорение	Эффективность
1	0.046000	1.0000	1.0000
2	0.072000	0.6388	0.3194
3	0.095000	0.4842	0.1614
4	0.100000	0.4600	0.1150
6	0.168000	0.2738	0.0456
9	0.206000	0.2233	0.0248
12	0.272000	0.1691	0.0140
18	0.386000	0.1191	0.0066
36	0.795000	0.0578	0.0016
99	2.327000	0.0197	0.0001

Как видно из результатов при увелечение количества потоков ускорение и эфффективность очень сильно падает, поэтому на моем компьютере выгоднее считать 1 потоком. Такие результаты получаются из-за того, что я работают на виртуальном компьютере.

Strace

```
vaney@V-box:~/Examples/os_lab3$ strace ./main 2
execve("./main", ["./main", "2"], 0x7ffd0a48f078 /* 58 vars */) =
0brk(NULL)                                          = 0x56356f3de000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe37a08210) = -1 EINVAL
```

```
(Invalidargument)
access("/etc/ld.so.preload", R_OK)              = -1 ENOENT (No
such file ordirectory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=67999, ...}) = 0 mmap(NULL,
67999, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fc2d02fa000
close(3)                                        = 0
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\0\220\201\0\0\0\0\0\0"..., 832) = 832
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O\305\3743\364B\2216\244\224\306@\
261\23\327o"..., 68, 824) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=157224, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fc2d02f8000
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O\305\3743\364B\2216\244\224\306@\
261\23\327o"..., 68, 824) = 68
mmap(NULL, 140408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fc2d02d5000
mmap(0x7fc2d02dc000, 69632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x7000) = 0x7fc2d02dc000
mmap(0x7fc2d02ed000, 20480, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3, 0x18000) = 0x7fc2d02ed000
mmap(0x7fc2d02f2000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1c000) = 0x7fc2d02f2000
mmap(0x7fc2d02f4000, 13432, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fc2d02f4000
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\3\0>\0\1\0\0\0\360q\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\20\0\0\0\5\0\0\0GNU\
0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\
355Y\377\t\334"..., 68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=2029224, ...}) = 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
pread64(3, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\
0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\363\377?\332\200\270\27\304d\245n\
355Y\377\t\334"..., 68, 880) = 68
mmap(NULL, 2036952, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fc2d00e3000
mprotect(0x7fc2d0108000, 1847296, PROT_NONE) = 0
mmap(0x7fc2d0108000, 1540096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x25000) = 0x7fc2d0108000
mmap(0x7fc2d0280000, 303104, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3, 0x19d000) = 0x7fc2d0280000
mmap(0x7fc2d02cb000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7fc2d02cb000
mmap(0x7fc2d02d1000, 13528, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fc2d02d1000
close(3)                                        = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fc2d00e0000
arch_prctl(ARCH_SET_FS, 0x7fc2d00e0740) = 0
```

```
mprotect(0x7fc2d02cb000, 12288, PROT_READ) = 0
 mprotect(0x7fc2d02f2000, 4096, PROT_READ) = 0
 mprotect(0x56356f031000, 4096, PROT_READ) = 0
 mprotect(0x7fc2d0338000, 4096, PROT_READ) = 0
munmap(0x7fc2d02fa000, 67999)                   = 0
set_tid_address(0x7fc2d00e0a10)                 = 3837
set_robust_list(0x7fc2d00e0a20, 24)             = 0
rt_sigaction(SIGRTMIN, {sa_handler=0x7fc2d02dcbf0,
sa_mask=[],
sa_flags=SA_RESTORER|SA_SIGINFO, sa_restorer=0x7fc2d02ea3c0}, NULL, 8) = 0
rt_sigaction(SIGRT_1, {sa_handler=0x7fc2d02dcc90, sa_mask=[],
sa_flags=SA_RESTORER|SA_RESTART|SA_SIGINFO, sa_restorer=0x7fc2d02ea3c0},
NULL, 8) = 0
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) =
0prlimit64(0, RLIMIT_STACK, NULL,
{rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) =
0brk(NULL)                                      = 0x56356f3de000
brk(0x56356f3ff000)                             = 0x56356f3ff000
write(1, "Thread number: 2\n", 17Thread number: 2
)               = 17
fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0
write(1, "Enter the number of degree of 1 "..., 44Enter the number of
degreeof 1 polynomial: ) = 44
read(0, 3
"3\n", 1024)                            = 2
read(0, 1 1 1
"1 1 1\n", 1024)                        = 6
write(1, "Enter the number of degree of 2 "..., 44Enter the number of
degreeof 2 polynomial: ) = 44
read(0, 5
"5\n", 1024)                            = 2
read(0, 1 2 3 4 5
"1 2 3 4 5\n", 1024)                    = 10
clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=2332129}) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7fc2cf8df000
mprotect(0x7fc2cf8e0000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7fc2d00defb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[3844], tls=0x7fc2d00df700,
child_tidptr=0x7fc2d00df9d0) = 3844
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7fc2cf0de000
mprotect(0x7fc2cf0df000, 8388608, PROT_READ|PROT_WRITE) = 0
clone(child_stack=0x7fc2cf8ddfb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[3845],
tls=0x7fc2cf8de700,child_tidptr=0x7fc2cf8de9d0) = 3845
clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=2649876}) =
0write(1, "Result: 1 3 6 9 12 9 5 \n", 24Result: 1 3 6 9 12 9 5
) = 24
write(1, "Execution time 0.317000 ms\n", 27Execution time 0.317000 ms
) = 27
lseek(0, -1, SEEK_CUR)                          = -1 ESPIPE (Illegal seek)
exit_group(0)                                   = ?
+++ exited with 0 +++
vaney@V-box:~/Examples/os_lab3$
```

```
*Бывает что выдает ошибку:
vaney@vaney-VirtualBox:~/OS/lab3$ ./main 2
Thread number: 2
Enter the number of degree of 1 polynomial: 5

1 1 1 1 1
Enter the number of degree of 2 polynomial: 2
2 3
main: malloc.c:2379: sysmalloc: Assertion `(old_top == initial_top (av) && old_size == 0)
|| ((unsigned long) (old_size) >= MINSIZE && prev_inuse (old_top) && ((unsigned long)
old_end & (pagesize - 1)) == 0)' failed.
Aborted (core dumped)
Но я долго думал, что это может быть, но так и не понял.
```

Вывод

Выполняя данную лабораторную работу, я научился работать с потоками с использованием POSIZ threads, а так же избежал проблем, связанных с критичной областью памяти благодаря мьютексу. Использование потоковбля решении задач может достаточно хорошо так усилить программу(многопоточность ускоряет работу программы), но составлять такую схему нелегко.