Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу

«Операционные системы»

Управление потоками в ОС. Обеспечение синхронизации между потоками.

Студент: Аминов Степан Сергеевич
Группа: М80 – 308Б
Вариант: 19
Преподаватель: Миронов Евгений Сергеевич
Дата:
Оценка:
Подпись: _____

Москва, 2021

1 Постановка задачи

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска программы.

Необходимо уметь продемонстрировать количество потоков, используемых программой, с помощью стандартных средств операционной системы.

Привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Объяснить получившиеся результаты.

Необходимо реализовать проверку числа на простоту при помощи алгоритма «решето Эратосфена».

2 Общий метод и алгоритм решения

При запуске программы у пользователя запрашивается число, которое необходимо проверить на простоту. Проверить на простоту можно только неотрицательное число, при вводе текста будет выведено сообщение о некорректном вводе и программа прекратит работу.

Из аргументов командной строки берётся количество потоков, которое может использовать программа. Производится выделение памяти для массива потоков, для массива аргументов потоковой функции и для самого решета. Решето представляет собой массив символов `sieve`. `sieve[i]` равно нулю, если число простое и единице в противном случае.

По определению числа 0 и 1 не являются простыми, поэтому сразу помечаем их единицами в решете. Необходимо проверить все числа от 2 до `num` включительно. Если ячейка решета, соответствующая числу `i`, равна нулю, то это число простое и требуется «вычеркнуть» (пометить единицей) все числа, кратные `i`. Эта задача и делегируется другим потокам.

Потоковая функция sieve_step принимает на вход число i и помечает единицами все числа, кратные `i`. Заметим, что первое число, кратное `i` и которое еще НЕ было помечено единицей – это чило i^2. Для ускорения алгоритма начнём проверку именно с этого числа и будем помечать каждое `i`-ое число, начиная с i^2. По этой же причине в главной функции перебор элементов решета будет вестись от 2 до корня из `i`. Потоки не смогут повлиять на работу друг друга, поэтому mutex не используется.

Укажем правило, по которому будет выбираться поток для выполнения функции. Заведём переменную `cur_thread`, изначально равную нулю. Для выполнения функции будет создаваться поток с индексом `cur_thred(mod threads_num)`, где `threads_num` – общее количество потоков. Таким образом, потоки будут использоваться в порядке закольцованной очереди. Когда `cur_thread` становится больше количества потоков, потоки начинают использоваться повторно. Во избежание ситуации, когда задача будет делегирована потоку, работа которого еще не окончена, будем дожидаться окончания работы потока. После делегирования задач, переменная `cur_thread` инкрементируется.

После обработки всего решета необходимо дождаться окончания работы всех активных потоков. После этого необходимо посмотреть число в `sieve[num]` и сделать вывод о простоте этого числа.

3 Основные файлы программы

```
#include <stdbool.h>

char* sieve;
long long num;
char c;
char lego[100];
int m = 0;

void* sieve_step(void* i_void) {
   long long i = *(long long*)i_void;
   for (long long j = i * i; j <= num; j += i) {
      sieve[j] = 1;
   }
   pthread_exit(NULL);
}

int main(int argc, char* argv[]) {
   if (argc != 2) {
      printf("./lab3 + число потоков\n");
      exit(1);
   }

   int threads_num = atoi(argv[1]);

   pthread_t* threads = (pthread_t*)calloc(threads_num, sizeof(pthread_t));

   long long* args = (long long*)malloc(threads_num * sizeof(long long));

   printf("Введите число для проверки: ");

   while (true) { // небольшой парсер
      c = getchar();
      if (c == '\n') {
         lego[m] = 0;
         num = strtol(lego, (char **) NULL, 10);
         break;
      } else if (c == ' '){
         lego[m] = 0;
         num = strtol(lego, (char **) NULL, 10);
         break;
      } else if (c >= '0' && c <= '9') {
         lego[m] = c;
         m++;
      } else{
         printf("Неверный ввод\n");
         return 0;
      }
```

```
    }


    sieve = (char*)calloc((num + 1), sizeof(char));

    sieve[0] = 1;
    sieve[1] = 1;

    int cur_thread = 0;
    for (long long i = 2; i * i <= num; ++i) {
        if (sieve[i] == 1) {
            continue;
        }
        if (cur_thread >= threads_num) {
            pthread_join(threads[cur_thread % threads_num], NULL);
        }

        args[cur_thread % threads_num] = i;
        pthread_create(&threads[cur_thread % threads_num], NULL, sieve_step,
&args[cur_thread % threads_num]);
        ++cur_thread;
    }

    for (int i = 0; i < threads_num; ++i) {
        pthread_join(threads[i], NULL);
    }

    if (sieve[num] == 1) {
        printf("%lld не простое число\n", num);
    }
    else {
        printf("%lld простое число\n", num);
    }

    free(sieve);
    free(threads);
    free(args);
}
```

4 Демонстрация работы программы

```
magic@magical:~/CLionProjects/os3$ ./lab3 1
Введите число для проверки: 151
151 простое число
magic@magical:~/CLionProjects/os3$ ./lab3 1
Введите число для проверки: 152
152 не простое число
magic@magical:~/CLionProjects/os3$ ./lab3
./lab3 + число потоков
magic@magical:~/CLionProjects/os3$ ./lab3 1
Введите число для проверки: -2
Неверный ввод
magic@magical:~/CLionProjects/os3$ ./lab3 1
Введите число для проверки: hey
Неверный ввод
magic@magical:~/CLionProjects/os3$ cat test
151476
magic@magical:~/CLionProjects/os3$ time ./lab3 1
Введите число для проверки: 999999
999999 не простое число


real   0m1,713s
user   0m0,035s
sys    0m0,009s
magic@magical:~/CLionProjects/os3$ time ./lab3 5
Введите число для проверки: 999999
999999 не простое число


real   0m1,451s
user   0m0,038s
sys    0m0,009s
magic@magical:~/CLionProjects/os3$ time ./lab3 55
Введите число для проверки: 999999
999999 не простое число


real   0m1,467s
user   0m0,032s
sys    0m0,023s
magic@magical:~/CLionProjects/os3$ strace -f -e trace="%process,write" -o
strace_log.txt ./lab3 1
Введите число для проверки: 151
151 простое число
magic@magical:~/CLionProjects/os3$ cat strace_log.txt
5376  execve("./lab3", ["./lab3", "1"], 0x7ffd32914760 /* 68 vars */) = 0
5376  arch_prctl(0x3001 /* ARCH_??? */, 0x7ffcee536b70) = -1 EINVAL (Invalid argument)
5376  arch_prctl(ARCH_SET_FS, 0x7f33e698d740) = 0
5376  write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \
321\207\320\270\321\201\320\273\320\276 \320\264\320\273\321\217"..., 51) = 51
5376  clone(child_stack=0x7f33e698bfb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[5377], tls=0x7f33e698c700,
child_tidptr=0x7f33e698c9d0) = 5377
5377  exit(0)                           = ?
5377  +++ exited with 0 +++
5376  clone(child_stack=0x7f33e698bfb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[5378], tls=0x7f33e698c700,
child_tidptr=0x7f33e698c9d0) = 5378
5378  exit(0)                           = ?
5378  +++ exited with 0 +++
5376  clone(child_stack=0x7f33e698bfb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
```

```
CLONE_CHILD_CLEARTID, parent_tid=[5379], tls=0x7f33e698c700,
child_tidptr=0x7f33e698c9d0) = 5379
5379  exit(0)                           = ?
5379  +++ exited with 0 +++
5376  clone(child_stack=0x7f33e698bfb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[5380], tls=0x7f33e698c700,
child_tidptr=0x7f33e698c9d0) = 5380
5380  exit(0)                           = ?
5380  +++ exited with 0 +++
5376  clone(child_stack=0x7f33e698bfb0, flags=CLONE_VM|CLONE_FS|CLONE_FILES|
CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, parent_tid=[5381], tls=0x7f33e698c700,
child_tidptr=0x7f33e698c9d0) = 5381
5381  exit(0)                           = ?
5381  +++ exited with 0 +++
5376  write(1, "151 \320\277\321\200\320\276\321\201\321\202\320\276\320\265 \
321\207\320\270\321\201\320\273\320\276\n", 30) = 30
5376  exit_group(0)                     = ?
5376  +++ exited with 0 +++
```

5 Выводы

В данной лабораторной работе мною был реализован и исследован алгоритм проверки числа на простоту при помощи решета Эратосфена.Можно заметить что при использовании двух-трёх потоков можно получить выигрыш по времени, но при использовании большего количества потоков ускорение не будет большим, так как операционной системе приходится тратить больше времени на выделение памяти под потоки и на их регулирование. Также я заметил, что создание потоков происходит быстрее, чем создание процессов, а все потоки используют одну и ту же область данных. Поэтому многопоточность – один из способов ускорить обработку каких-либо данных: выполнение однотипных, не зависящих друг от друга задач, можно поручить отдельным потокам, которые будут работать парарллельно.