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: Дубровин Дмитрий
Группа: М80-207Б-21
Вариант: 18
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/1droozd1/os_labs/tree/main/lab_3

Постановка задачи

Цель работы

Приобретение практических навыков в:

1. Управление потоками в ОС
2. Обеспечение синхронизации между потоками

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.

Задание по варианту (вариант 18): Найти образец в строке наивным алгоритмом

Общие сведения о программе

Программа представляет из себя один файл main.c

Общий метод и алгоритм решения

Наивный алгоритм на то и наивный, что очень простой для использования и реализации. ПО факту это простой перебор определенного pattern в строке. Для ускорения данного действа я, выполняя поставленное задание, использую многопоточность. Я решил разбить исходную строку на множество подстрок, далее же получившиеся подстроки отправляются на обработку потоками. Для решения проблем доступа потоков к данным, я использовал структуру и выделение памяти с помощью malloc.

Исходный код

Main.c

```
#include "stdio.h"
```

```
#include "stdlib.h"
#include "pthread.h"
#include "string.h"
#include "time.h"

//valgrind --tool=memcheck ./a.out

typedef struct thread_data {

    long int size;
    char *pat;
    char *txt;

} thread_data;

pthread_mutex_t mutex;

void *threads_searching(void* args)
{
    thread_data *tdata = (thread_data *)args;

    char *pattern = tdata->pat;
    char *string_from_text = tdata->txt;
    int size = tdata -> size;

    int len_of_pattern = strlen(pattern);
    int len_of_str = strlen(string_from_text);

    int flag1 = 0;

    for (int i = 0; i < len_of_str - len_of_pattern; i++) {
        for (int j = 0; j < len_of_pattern; j++) {
            if (string_from_text[i + j] != pattern[j]) {
                break;
            }
```

```
            if (j == len_of_pattern - 1) {
                int size1 = i + size;
                printf("Pattern found at index: %d\n", size1);
                flag1 = 1;
            }
        }
    }

    free(tdata);
    free(string_from_text);
    return NULL;
}


int main(int argc, char *argv[])
{
    pthread_t *th;
    int threads_amount = atoi(argv[0]);

    if (threads_amount < 2) {
        printf("Write amount of threads: ");
        scanf("%d", &threads_amount);
    }
    printf("Amount of threads = %d\n", threads_amount);

    int len_txt;
    printf("Write amount of symbols in text: ");
    scanf("%d", &len_txt);

    char *text;
    text = (char*) malloc(len_txt * sizeof(char));

    for (int i = 0; i < len_txt; i++) {
        char randomletter = "ABC"[random () % 3];
        text[i] = randomletter;
```

```
}

//printf("text = %s\n", text);

char pat[20];
printf("Write pattern - ");
scanf("%s", pat);

int lenght_pat = strlen(pat);

while (lenght_pat > len_txt) {
    printf("Wrong pattern, write pattern less than a string: \n");
    scanf("%s", pat);
    printf("Pattern = %s\n", pat);
}

clock_t t1, t2;

t1 = clock();

int max_threads_amount = len_txt / lenght_pat;

if (threads_amount > max_threads_amount) {
    threads_amount = max_threads_amount;
    printf("Too many threads, max amount of threads is %d\n", threads_amount);
}

th = (pthread_t *) malloc(threads_amount * sizeof(pthread_t));

if (th == NULL) {
    printf("Error with threads\n");
}

int kolSym_in_str = (len_txt / threads_amount);
char *strok;
```

```
int flag1 = 0;

if (len_txt % threads_amount != 0) {
   flag1 = 1;
}

int j = 0;

for (int i = 0; i < threads_amount; i++) {

   thread_data *tdata = malloc(sizeof(thread_data));

   if (flag1 == 1) { // the number of text characters is not
                     //divisible by the number of threads

      if (i == (threads_amount - 1)) {

         strok = (char*) malloc((kolSym_in_str + (len_txt % threads_amount)) * (sizeof(char)));
         strncpy(strok, (char *) &text[j], kolSym_in_str + (len_txt % threads_amount));

      } else {

         strok = (char*) malloc((kolSym_in_str + lenght_pat - 1) * (sizeof(char)));
         strncpy(strok, (char *) &text[j], kolSym_in_str + lenght_pat - 1);

      }

   } else { //the number of characters divided by the number of threads (without modulo)

      if (i == (threads_amount - 1)) {

         strok = (char*) malloc((kolSym_in_str) * sizeof(char));
         strncpy(strok, (char *) &text[j], kolSym_in_str);
```

```
            } else {

                strok = (char*) malloc((kolSym_in_str + lenght_pat - 1) * (sizeof(char)));
                strncpy(strok, (char *) &text[j], kolSym_in_str + lenght_pat - 1);

            }
        }

        tdata -> pat = (char *)pat;
        tdata -> txt = (char *)strok;
        tdata -> size = j;

        j += kolSym_in_str;

        if ((pthread_create(&th[i], NULL, threads_searching, (void *)tdata)) != 0) {
            perror("Failed to create thread");
        }
    }

    for (int i = 0; i < threads_amount; i++) {
        if ((pthread_join(th[i], NULL)) != 0) {
            perror("Failed to join thread");
        }
    }

    free(text);
    free(th);

    t2 = clock();

    printf("Compilation time: %f\n", (t2 - t1) / (double)CLOCKS_PER_SEC);

    return 0;
}
```

Демонстрация работы программы

dr0ozd1@Dmitry-Nitro-AN515-45:~/Coding/Lab_os/os_labs/lab_3/src$ gcc main.c

dr0ozd1@Dmitry-Nitro-AN515-45:~/Coding/Lab_os/os_labs/lab_3/src$ time ./a.out

Write amount of threads: 16

Amount of threads = 16

Write amount of symbols in text: 10000000

Write pattern - ABCBABCBBA

Pattern found at index: 636956

Pattern found at index: 115396

Pattern found at index: 694665

Pattern found at index: 1300079

Pattern found at index: 139318

Pattern found at index: 1906209

Pattern found at index: 1320346

Pattern found at index: 2556116

Pattern found at index: 1968164

Pattern found at index: 1388360

Pattern found at index: 2613310

Pattern found at index: 4396172

Pattern found at index: 846060

Pattern found at index: 3831335

…

Compilation time: 0.084684

real 0m9,631s

user 0m0,139s

sys 0m0,000s

С ипользованием strace:

dr0ozd1@Dmitry-Nitro-AN515-45:~/Coding/Lab_os/os_labs/lab_3/src$ gcc main.c

dr0ozd1@Dmitry-Nitro-AN515-45:~/Coding/Lab_os/os_labs/lab_3/src$ strace -f -e 'clone' ./a.out

Program pid: 48841

Write amount of threads: 16

Amount of threads = 16

Write amount of symbols in text: 10000000

Write pattern: ABCBACBBBA

strace: Process 48926 attached

strace: Process 48927 attached

Pattern found at index: 45515

strace: Process 48928 attached

Pattern found at index: 107515

Pattern found at index: 108433

Pattern found at index: 121800

strace: Process 48929 attached

strace: Process 48930 attached

Pattern found at index: 185857

Pattern found at index: 200286

strace: Process 48931 attached

Pattern found at index: 228728

Pattern found at index: 1389555

strace: Process 48932 attached

Pattern found at index: 3772274

Pattern found at index: 309906

strace: Process 48933 attached

…

Pattern found at index: 9856767

[pid 48934] +++ exited with 0 +++

Pattern found at index: 8734829

Pattern found at index: 9301622

Pattern found at index: 8110928

[pid 48939] +++ exited with 0 +++

Pattern found at index: 9341648

[pid 48938] +++ exited with 0 +++

[pid 48940] +++ exited with 0 +++

Pattern found at index: 9956748

[pid 48941] +++ exited with 0 +++

Compilation time: 0.088290

+++ exited with 0 +++

Выводы

При выполнении данной лабораторной работы я приобрел практические навыки работы с потоками в C. К тому же, я научился использовать различные способы для решения проблем возникающих при использовании многопоточного программирования.