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 "unistd.h"
#include "time.h"

//valgrind --tool=memcheck ./a.out
//strace -f -e 'clone' ./a.out

typedef struct thread_data {

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

} thread_data;



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

    char *pattern = tdata -> pat;
    long int count = tdata -> size;
    long int size = tdata -> kol_sym;
    char *text = tdata -> txt;

    char *string_from_text;
    string_from_text = (char *) malloc((size + 1) * sizeof(char));
    strncpy(string_from_text, (char *) &text[count], size);
    string_from_text[size] = '\0';



    int len_of_pattern = strlen(pattern);
```

```
    for (int i = 0; i < size - 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) {
                printf("Pattern found at index: %ld\n", count + 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;
}

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)) {

         tdata -> kol_sym = kolSym_in_str + (len_txt % threads_amount) + 1;

      } else {

         tdata -> kol_sym = kolSym_in_str + lenght_pat;

      }

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

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

         tdata -> kol_sym = kolSym_in_str + lenght_pat - 1;

      } else {
```

```
            tdata -> kol_sym = kolSym_in_str;
        }
    }

    tdata -> pat = (char *)pat;
    tdata -> txt = (char *)text;
    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

```
Write amount of threads: 16
Amount of threads = 16
Write amount of symbols in text:
```

1000000

Write pattern: ABCBABBAB

Pattern found at index: 1

Pattern found at index: 187501

Pattern found at index: 312501

Pattern found at index: 62501

Pattern found at index: 250001

Pattern found at index: 62501

Pattern found at index: 312501

Pattern found at index: 500001

…

Pattern found at index: 437501

Pattern found at index: 250001

Pattern found at index: 62501

Pattern found at index: 812501

Pattern found at index: 312501

Pattern found at index: 625001

Pattern found at index: 500001

Pattern found at index: 937501

Pattern found at index: 562501

Pattern found at index: 812501

Pattern found at index: 500001

Pattern found at index: 437501

Pattern found at index: 625001

Pattern found at index: 750001

Pattern found at index: 937501

Pattern found at index: 750001

Pattern found at index: 562501

Pattern found at index: 937501

Pattern found at index: 562501

Pattern found at index: 875001

Pattern found at index: 875001

Pattern found at index: 875001

Pattern found at index: 875001

Pattern found at index: 875001

Compilation time: 0.010309

Amount of symbols in text ->	1 million	10 million	50 million	100 million
1 thread	0m0,013s	0m0,113s	0m0,559s	0m1,142s
16 threads	0m0,008s	0m0,062s	0m0,298s	0m0,594s
32 threads	0m0,008s	0m0,064s	0m0,296s	0m0,622s

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.