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

Лабораторная работа №3 по курсу

«Операционные системы»

Тема работы

“Потоки”

Студент: Каширин Кирилл Дмитриевич
Группа: М8О-208Б-20
Вариант: 5
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/lumses/OS

Постановка задачи

Программа должна обрабатывать данные в многопоточном режиме, используя стандартные средства создания потоков операционной системы Unix. Ограничение потоков должно быть задано ключом запуска программы.

Задание. Отсортировать массив целых чисел при помощи четно-нечетной сортировки Бетчер.

Общие сведения о программе

Программа написана на языке Си в UNIX-подобной операционной системе.

Для компиляции требуется указать ключ –pthread. Для запуска программы в качестве первого аргумента командной строки необходимо указать количество потоков.

Сборка проекта происходит при помощи make-файла

gcc –pthread os_lab3.c

Общий метод и алгоритм решения

Зная количество потоков, мы распределяем каждому потоку диапазон в массиве, который он будет сортировать. Все эти данные хранятся в структуре tsk (номера поток, нижняя граница сортировки, верхняя граница сортировки). Если потоков больше чем размер массива, то мы используем количество поток равное размеру массива, если при делении размер массива на количество потоков получается нецелое число, то последний поток сортирует часть массива с остатком. Чтобы отсортировать массив, используется четно-нечетная сортировка Бетчер. Смысл этой сортировки в том, что изначально в начало массива переставляются элементы, которые стояли на четных позициях, в конец массива – на нечетных позициях. После этого используется сортировка слиянием(merge_sort).

В программе, используются две функции, связанные с потоками:

pthread_create – создание потока.

pthread_join(pthread_t thread, void **value_ptr)- Откладывает выполнение вызывающего (эту функцию) потока, до тех пор, пока не будет выполнен поток thread.

Исходный код

os_lab3.c

```
#include <stdio.h>
#include <pthread.h>
#include <time.h>
#include <stdlib.h>
int *a;
struct tsk {
    int number;
    int l;
    int r;
};
void shuffle(int size, int low, int high){
    int mid = (low + high) / 2;
    int* p = (int*) malloc(30 * sizeof(int));
    for (int i = low, j = 0; i <= high; i += 2, j++) {
        if (low + j <= high) {
            p[i] = a[low + j];
        }
        if (mid + j + 1 <= high) {
            p[i + 1] = a[mid + j + 1];
        }
    }
    for (int i = 0; i < size; i++) {
        a[i]  = p[i];
        //  printf("s%d ",a[i]);}
        //  printf("\n");
    }
    free(p);
}
void merge(int low, int mid, int high) {
    int *left = malloc((mid-low+1)*sizeof(int));
    int *right = malloc((high-mid)*sizeof(int));
    for (int i = 0; i < mid-low+1; i++) {
        left[i] = a[i + low];
    }
    for (int i = 0; i < high-mid; i++) {
        right[i] = a[i + mid + 1];
    }
    int k = low;
    int i = 0, j = 0;
    while (i < mid-low+1 && j < high-mid) {
```

```
            if (left[i] <= right[j]) {
                a[k++] = left[i++];
            } else {
                a[k++] = right[j++];
            }
        }
        while (i < mid-low+1) {
            a[k++] = left[i++];
        }
        while (j < high-mid) {
            a[k++] = right[j++];
        }
        free(left);
        free(right);
}
void merge_sort(int low, int high) {
        int mid = (low + high) / 2;
        if (low < high) {
            merge_sort(low, mid);
            merge_sort(mid + 1, high);
            merge(low, mid, high);
        }
}
void* merge_sort_thread(void* arg) {
        struct tsk *tsk = arg;
        int mid = (tsk->l + tsk->r) / 2;
        if (tsk->l < tsk->r) {
            merge_sort(tsk->l, mid);
            merge_sort(mid + 1, tsk->r);
            merge(tsk->l, mid, tsk->r);
        }
}
int main(int argc, char *argv[]) {
        struct tsk* control_param;
        int size, number_threads;
        printf("Number elements:");
        scanf("%d",&size);
        // printf("%c",*argv[1]);
        number_threads= *argv[1];
        //printf("Number threads:");
```

```
    //scanf("%d",&number_threads);
    a = malloc(sizeof(int) * size);
    for (int i=0;i<size;i++) {
        scanf("%d",&a[i]);
    }
    shuffle(size,0,size-1);
    pthread_t threads[number_threads];
    struct tsk tsklist[number_threads];
    if (number_threads > size) {
        number_threads == size;
    }
    int len = size/number_threads;
    int low = 0;
    for (int i = 0; i < number_threads; i++, low += len) {
        control_param = &tsklist[i];
        control_param->number = i;
        control_param->l = low;
        control_param->r = low + len - 1;
        if (i == (number_threads-1)) {
            control_param->r = size - 1;
        }
    }
    for (int i = 0; i < number_threads; i++) {
        control_param = &tsklist[i];
        pthread_create(&threads[i], NULL, merge_sort_thread,
control_param);
    }
    for (int i = 0; i < number_threads; i++) {
        pthread_join(threads[i], NULL);
    }
   /* for (int i = 0; i < number_threads; i++) {
        control_param = &tsklist[i];
        printf("SUB %d:", control_param->number);
        for (int j = control_param->l; j <= control_param->r;
++j)
            printf(" %d", a[j]);
            printf("\n");
        } */
    struct tsk *tskm = &tsklist[0];
    for (int i = 1; i < number_threads; i++) {
```

```
        struct tsk *tsk = &tsklist[i];
        merge(tskm->l, tsk->l - 1, tsk->r);
    }
    printf("Sorted array:");
    for (int i = 0; i < size; i++)  {
        printf(" %d", a[i]);
    }
    printf("\n");
    return 0;
}
```

Демонстрация работы программы

```
kirill@LAPTOP-F153AKTP:~/OS/os_lab3/src$ make
gcc -pthread os_lab3.c
kirill@LAPTOP-F153AKTP:~/OS/os_lab3/src$ ./a.out 5
Number elements:5
2 34 5 1 2
Sorted array: 1 2 2 5 34
kirill@LAPTOP-F153AKTP:~/OS/os_lab3/src$ ./a.out 10
Number elements:8
11 2 18 9 1 0 2 5
Sorted array: 0 1 2 2 5 9 11 18
kirill@LAPTOP-F153AKTP:~/OS/os_lab3/src$ ./a.out 1
Number elements:4
3 2 18 9
Sorted array: 2 3 9 18
```

Выводы

Эта лабораторная работа ознакомила и научила меня работать с потоками. Я изучил основные функции для работы с потоками в Си, как совершать системные запросы на создание потока, ожидание завершения потока. Я изучал понятие “многопоточность”. Она позволяет ускорить обработку данных в программе