***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:*** Шандрюк Пётр Николаевич

***Группа: М8О-208Б-20***

***Вариант: 3***

***Преподаватель: Миронов Евгений Сергеевич***

***Оценка: \_\_\_\_\_\_\_\_\_\_\_***

***Дата: \_\_\_\_\_\_\_\_\_\_\_***

***Подпись: \_\_\_\_\_\_\_\_\_\_\_***

***Москва, 2021***

***Содержание***

1. ***Репозиторий***

***2. Постановка задачи***

***3. Общие сведения о программе***

***4. Общий метод и алгоритм решения***

***5. Исходный код***

***6. Демонстрация работы программы***

***7. Выводы***

***Репозиторий***

https://github.com/Peter1811/OS

***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.

3 вариант) Многопоточная сортировка слиянием.

***Общие сведения о программе***

Программа компилируется из файла sort.c. Также используется библиотеки: pthread.h, time.h, stdlib.h. В программе используются следующие функции:

1. ***Pthread\_create - создает задачу для выполнения в потоке.***
2. ***Pthread\_join - объединяет результаты задач из потоков.***

***Общий метод и алгоритм решения***

Для реализации поставленной задачи необходимо:

1. ***Изучить принципы работы pthread\_create, pthread\_join.***
2. ***Написать программу, которая будет работать с потоками. Организовать работу с созданием n потоков (число n передается аргументом командной строки).***

***Исходный код***

***sort.c***

#include<stdio.h>

#include<pthread.h>

#include<time.h>

#include<stdlib.h>

int\*a;

Struct tsk{

intnumber;

intl;

intr;

};

Void merge(int low, int mid, int high) {

int\*left=malloc((mid-low+1)\*sizeof(int));

int\*right=malloc((high-mid)\*sizeof(int));

for(inti=0; i<mid-low+1; i++) {

left[i] =a[i+low];

}

for(inti=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);

}

voidmerge\_sort(intlow, inthigh) {

intmid=(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){

structtsk\*tsk=arg;

intmid=(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);

}

}

intmain(intargv, char\*argc[]) {

structtsk\*control\_param;

intsize, number\_threads;

printf("Number elements:");

scanf("%d",&size);

number\_threads=\*argc[1];

a=malloc(sizeof(int) \*size);

for(inti=0;i<size;i++){

scanf("%d",&a[i]);

}

pthread\_tthreads[number\_threads];

structtsktsklist[number\_threads];

if(number\_threads>size) {

number\_threads==size;

}

intlen=size/number\_threads;

intlow=0;

for(inti=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(inti=0; i<number\_threads; i++) {

control\_param=&tsklist[i];

pthread\_create(&threads[i], NULL, merge\_sort\_thread, control\_param);

}

for(inti=0; i<number\_threads; i++) {

pthread\_join(threads[i], NULL);

}

structtsk\*tskm=&tsklist[0];

for(inti=1; i<number\_threads; i++) {

structtsk\*tsk=&tsklist[i];

merge(tskm->l, tsk->l-1, tsk->r);

}

printf("Sorted array:");

for(inti=0; i<size; i++){

printf("%d", a[i]);

}

printf("\n");

return0;

}

***Демонстрация работы программы***

***[Temi4@localhost ~]$ cd*** /mnt/c/users/peter/desktop/os

***[Temi4@localhost 2\_lab]$ gcc sort.c***

***[Temi4@localhost 2\_lab]$* ./a.out 5**

***Number elements:5***

***3 425 6 78 6***

***Sorted array: 3 6 6 78 425***

***Выводы***

***В процессе данной работы были разобраны функции для работы с потоками, например - pthread\_create и pthread\_join. Также было рассморено разделение многопоточной сортировки на несколько потоков - при больших размерах массива это будет эффективнее по времени. Поняты общие принципы работы многопоточного программирования. С помощью утилиты strace мы можем отследить системный вызов clone, который свидетельствует о создании потока.***