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: Хомяков Иван Андреевич
Группа: М8О-207Б-21
Вариант: 17
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/EbumbaE/OS_LAB/lab3

Постановка задачи

Цель работы

Целью является приобретение практических навыков в:

- Управление потоками в ОС
- 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.

Вариант 17: найти в большом целочисленном массиве минимальный элемент.

Общие сведения о программе

Программа компилируется из файла main.c. Также используется заголовочный файл: main.h, который описывает функции и подключает include.h. main читает элементы массива. Отправляет два указателя на участок массива, который нужно проанализировать в отдельный поток.
В программе используются следующие системные вызовы:

1. _pipe – создает канал для общения main с потоками, принимает массив, который будет использоваться для хранения дескрипторов файлов чтения и записи, размер канала и файловый режим.
2. _beginthread – создает поток. Принимает адрес подпрограммы, размер стека подпрограммы и список аргументов.
3. _write – запись в файл, принимает дескриптор файла, буфер данный и размер буфера. Используется нами для записи в канал.
4. _read – читает из файла, принимает дескриптор файла, буфер данный и размер буфера. Используется нами для чтения из канала.
5. _close – закрывает файл, принимает дескриптор файла. Используется нами для закрытия канала.

Время работы зависит от количества потоков. Среднее время для 10 элементов и 5 потоков: 50. Среднее время для 10 элементов и 3 потоков: 18.6. Среднее время для 10 элементов и 1 потока: 9. Напишем программу, которая ищет минимальный элемент, просто пройдя по массиву, среднее время для 10 элементов: 2,8. Из этого мы делаем вывод, что чем меньше потоков мы запускаем для поиска минимального элемента массива, тем быстрее работает программа.

Общий метод и алгоритм решения

Прочитаем количество потоков и количество элементов массива. Если один поток будет обрабатывать меньше 2 элементов, то уменьшим количество потоков, заданных пользователем. Создаем канал. Пока вводим элементы вычисляем указатели на начало и конец массива. Перед вызовом потока закидываем в отдельную память указатели, чтобы только поток мог с ними работать. В потоке после вычислений освобождаем выделенную память. Последний поток запускаем вне цикла чтения, т.к. распределение элементов на потоки может быть неравномерное (10 элементов, 3 потока, последний поток в итоге получит 4 элемента на обработку). Читаем из канала все минимумы, найденные потоками и среди них вычисляем ответ. Выводим его, закрываем канал. В данной программе потоков дожидаться не нужно, тк мы точно знаем сколько мы их создали и в цикле в итоге получим от каждого значение, а после этого выйдем из main. Потоки не убегут)

Исходный код

В репозитории.

Демонстрация работы программы

```
Input amount treads: 3
Input amount elements: 10
1 2 3 4 5 6 7 8 9 10
in 1 thread
in 2 thread
in 3 thread
tread min: 1
tread min: 4
tread min: 7

result: 1
all is ok
```

Выводы

Создал программу, потоки в которой, общаются с основным при помощи канала.