Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №6
по дисциплине ОСиСП

Тема: «Работа с файлами, отображенными в память»

Выполнил:
студент группы 150501
Ковальчук Д.И.

Проверил:
старший преподаватель
_____ Поденок Л. П.

Минск
2023

1 ПОСТАНОВКА ЗАДАЧИ

Кооперация потоков для высокопроизводительной обработки больших файлов. Изучаемые системные вызовы: `pthread_create()`, `pthread_exit()`, `pthread_join()`, `pthread_yield()`, `pthread_cancel()`, `pthread_barrier_init()`, `pthread_barrier_destroy()`, `pthread_barrier_wait()`, `mmap()`, `munmap()`.

Задание

Написать многопоточную программу `sort_index` для сортировки вторичного индексного файла таблицы базы данных, работающую с файлом в двух режимах: `read()`/`write()` и с использованием отображение файлов в адресное пространство процесса. Программа должна запускаться следующим образом:

```
sort_index memsize granul threads filename
```

Параметры командной строки:

`memsize` := размер рабочего буфера, кратный размеру страницы (`getpagesize()`)

`blocks` := порядок разбиения буфера

`threads` := количество потоков (от `k` до `N`)

`k` := количество ядер

`N` := максимальное количество потоков (`8k??`)

`filename` := имя файла

Количество блоков должно быть степенью двойки и превышать количество потоков.

Для целей тестирования написать программу генерации неотсортированного индексного файла.

2 АЛГОРИТМ

2.1 Алгоритм программы генерации

Генерируемый файл представляет собой вторичный индекс по времени и состоит из заголовка и индексных записей фиксированной длины. Индексная запись имеет следующую структуру:

```
struct index_s {
```

```
        // временная метка (модифицированная юлианская дата)
        double time_mark;
        uint64_t recno; // первичный индекс в таблице БД
    } index_record;
```

Заголовок представляет собой следующую структуру

```
    struct index_hdr_s {
        uint64_t records; // количество записей
        struct index_s idx[]; // массив записей в количестве records
    }
```

Временная метка определяется в модифицированный юлианских днях. Целая часть лежит в пределах от 15020.0 (1900.01.01-0:0:0.0) до «вчера». Дробная – это часть дня (0.5–12:0:0.0). Для генерации целой и дробной частей временной метки используется системный генератор случайных чисел (`random(3)`).

Первичный индекс, как вариант, может заполняться последовательно, начиная с 1, но может быть случайным целым > 0 (в программе сортировки не используется). Размер индекса в записях должен быть кратен 256 и кратно превышать планируемую выделенную память для отображения. Размер индекса и имя файла указывается при запуске программы генерации.

2.2 Алгоритм программы сортировки

1) Основной поток запускает `threads` потоков, сообщая им адрес буфера, размер блока `memsize/blocks`, и их номер от 1 до `threads - 1`, используя возможность передачи аргумента для `start_routine`. Порожденные потоки останавливаются на барьере, ожидая прихода основного.

2) Основной поток с номером 0 открывает файл, отображает его часть размером `memsize` на память и синхронизируется на барьере. Барьер «открывается» и все `threads` потоков входят на равных в фазу сортировки.

3) Фаза сортировки

С каждым из блоков связана карта (массив) отсортированных блоков, в которой изначально блоки с 0 по `threads - 1` отмечены, как занятые.

Поток `n` начинает с того, что выбирает из массива блок со своим номером и его сортирует, используя `qsort(3)`. После того,

как поток отсортировал свой первый блок, он на основе конкурентного захвата мьютекса, связанного с картой, получает к ней эксклюзивный доступ, отмечает следующий свободный блок, как занятый, освобождает мьютекс и приступает к его сортировке.

Если свободных блоков нет, синхронизируется на барьере. После прохождения барьера все блоки будут отсортированы.

4) Фаза слияния

Поскольку блоков степень двойки, слияния производятся парами в цикле.

Поток 0 сливает блоки 0 и 1, поток 1 блоки 2 и 3, и так далее.

Для отметки слитых пар и не слитых используется половина карты. Если для потока нет пары слияния, он синхронизируется на барьере.

В результате слияния количество блоков, подлежащих слиянию сокращается в два раза, а размер их в два раза увеличивается.

После очередного прохождения барьера количество блоков, подлежащих слиянию, станет меньше количества потоков. В этом случае распределение блоков между потоками осуществляется на основе конкурентного захвата мьютекса, связанного с картой. Потоки, котором не досталось блока, синхронизируются на барьере.

Когда осталась последняя пара, все потоки с номером не равным нулю синхронизируются на барьере, о поток с номером 0 выполняет слияние последней пары.

После слияния буфер становится отсортирован и подлежит сбросу в файл (`munmap()`).

Если не весь файл обработан, продолжаем с шага 2).

Если весь файл обработан, основной поток отправляет запрос отмены порожденным потокам, выполняет слияние отсортированных частей файла и завершается.

3 ТЕСТ

```
darya@~/osisp/lab06$ sh script.sh

file.bin CREATED AND FILLED WITH DATA

AMOUNT OF RECORDS: 4096

DATE IS 16815.258671, INDEX IS 1
DATE IS 18949.206731, INDEX IS 2
DATE IS 16507.644442, INDEX IS 3
DATE IS 18996.421665, INDEX IS 4
DATE IS 17460.783876, INDEX IS 5
```

```
DATE IS 17954.067313, INDEX IS 6
DATE IS 17979.521010, INDEX IS 7
DATE IS 17762.915834, INDEX IS 8
DATE IS 18343.713781, INDEX IS 9
DATE IS 15743.581391, INDEX IS 10
DATE IS 15998.631871, INDEX IS 11
…
DATE IS 18788.177778, INDEX IS 4085
DATE IS 17868.449730, INDEX IS 4086
DATE IS 19348.672930, INDEX IS 4087
DATE IS 16385.782000, INDEX IS 4088
DATE IS 17621.656559, INDEX IS 4089
DATE IS 19296.935983, INDEX IS 4090
DATE IS 16748.337533, INDEX IS 4091
DATE IS 17909.941701, INDEX IS 4092
DATE IS 18790.884654, INDEX IS 4093
DATE IS 16244.625686, INDEX IS 4094
DATE IS 16974.877005, INDEX IS 4095
DATE IS 16890.721929, INDEX IS 4096


file.bin SORTED AND SAVED TO sorted_file.bin

DATE IS 15021.775496, INDEX IS 1099
DATE IS 15021.796055, INDEX IS 2508
DATE IS 15024.390743, INDEX IS 2607
DATE IS 15024.509686, INDEX IS 214
DATE IS 15025.432775, INDEX IS 319
DATE IS 15025.495017, INDEX IS 535
DATE IS 15029.525284, INDEX IS 3277
DATE IS 15030.393434, INDEX IS 586
DATE IS 15032.215369, INDEX IS 1887
DATE IS 15033.757567, INDEX IS 4047
DATE IS 15037.326358, INDEX IS 111
DATE IS 15037.330438, INDEX IS 1469
…
DATE IS 19494.340608, INDEX IS 2382
DATE IS 19497.446567, INDEX IS 3241
DATE IS 19499.504033, INDEX IS 3799
DATE IS 19501.535507, INDEX IS 1755
DATE IS 19502.802858, INDEX IS 1577
DATE IS 19504.278032, INDEX IS 3080
DATE IS 19504.918983, INDEX IS 921
DATE IS 19506.362989, INDEX IS 3328
DATE IS 19506.876912, INDEX IS 1628
DATE IS 19507.099282, INDEX IS 1335
DATE IS 19508.052872, INDEX IS 2948
DATE IS 19509.190031, INDEX IS 847

AMOUNT OF RECORDS 4096
-------------------------------------------------------------------
```

4 ВЫВОД

В ходе лабораторной работы была изучена работа с файлами, отображенными в память. При выполнении лабораторной работы были использованы следующие системные вызовы: `pthread_create()`, `pthread_exit()`, `pthread_join()`, `pthread_yield()`, `pthread_cancel()`, `pthread_barrier_init()`, `pthread_barrier_destroy()`, `pthread_barrier_wait()`, `mmap()`, `munmap()`.