Министерство Образования и Науки Российской Федерации

Федеральное Государственное Бюджетное Образовательное Учреждение Высшего Профессионального Образования «Санкт-Петербургский Национальный Исследовательский Университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по дисциплине

“Операционные системы”

Выполнил:

Студент группы P3330,

Тарасов Иван Сергеевич

Проверил:

Романов Артемий Ильич

 Санкт-Петербург

2024

**Задание**

Необходимо реализовать блочный кэш в пространстве пользователя в виде динамической библиотеки (dll или so). Политику вытеснения страниц – **Optimal**.

При выполнении работы необходимо реализовать простой API для работы с файлами, предоставляющий пользователю следующие возможности:

1. Открытие файла по заданному пути файла, доступного для чтения. Процедура возвращает некоторый хэндл на файл.

int lab2\_open(const char \*path).

1. Закрытие файла по хэндлу.

int lab2\_close(int fd).

1. Чтение данных из файла.

ssize\_t lab2\_read(int fd, void buf[.count], size\_t count).

1. Запись данных в файл.

ssize\_t lab2\_write(int fd, const void buf[.count], size\_t count).

1. Перестановка позиции указателя на данные файла.  
   ​​​off\_t lab2\_lseek(int fd, off\_t offset, int whence).
2. Синхронизация данных из кэша с диском.

int lab2\_fsync(int fd).

1. Подсказка для API, которая определяет, через какое время, разрешено будет вытеснять страницы

access\_hint\_t lab2\_advice(uint64\_t inode, access\_hint\_t hint).

Операции с диском разработанного блочного кеша должны производиться в обход page cache используемой ОС.

В рамках проверки работоспособности разработанного блочного кэша необходимо адаптировать указанную преподавателем программу-загрузчик из ЛР 1, добавив использование кэша. Запустите программу и убедитесь, что она корректно работает. Сравните производительность до и после.

Код программы с пояснениями загружен на GitHub:

[diy-page-cache-IoannTar2004/src/cacheAPI/cache.cpp at lab2 · secs-dev-os-course/diy-page-cache-IoannTar2004](https://github.com/secs-dev-os-course/diy-page-cache-IoannTar2004/blob/lab2/src/cacheAPI/cache.cpp)

**Нагрузчик**

Нагрузчик в данной лабораторной работе выполняет алгоритм поиска подстроки в файле. Для того, чтобы продемонстрировать работу кэша, требуется нагрузить подсистему ввода-вывода. Для этого программа производит чтение каждого символа. Программа выполнила данный алгоритм 10 раз.

Сравнение времени работы алгоритма (чтение файла размером 8 Мбайт)

|  |  |  |
| --- | --- | --- |
| **№** | **В обход кэша** | **С моим кэшем** |
| 1 | 1s 141253912ns | 4s 474075664ns |
| 2 | 1s 111265919ns | 1s 116752025ns |
| 3 | 1s 138215166ns | 1s 74435182ns |
| 4 | 1s 149081608ns | 1s 90631644ns |
| 5 | 1s 138343964ns | 1s 62332350ns |
| 6 | 1s 126686630ns | 1s 111724405ns |
| 7 | 1s 162710578ns | 1s 116571383ns |
| 8 | 1s 157388987ns | 1s 85354314ns |
| 9 | 0s 859123571ns | 1s 103346865ns |
| 10 | 1s 142675501ns | 0s 893261331ns |

**Выводы**

По таблице сравнения времени работы алгоритма можно заметить, что время работы в случае обхода кэша практически одинаковое и в среднем занимает 1,5 секунды. Однако в случае работы с записью в мой кэш, время работы на каждой итерации столько же, сколько и в случае с обходом кэша, при этом на первой итерации чтение происходит значительно дольше – почти 4,5 секунды. Возможно, флаг O\_DIRECT не гарантирует напрямую чтение и запись из файла в обход блочного кэша моей ОС. С другой стороны, время работы с моим кэшем в последующих итерациях заметно меньше, чем в первой, что означает, что мой кэш работает правильно.