Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин

Лабораторная работа №7
по курсу
«Операционные системы и системное программирование»
на тему
«Блокировки чтения/записи»

Выполнил: студент группы 350501
Слепица О.Н.
Проверил: старший преподаватель каф. ЭВМ
Поденок Л.П.

Минск 2025

1 ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

Конкурентный доступ к совместно используемому файлу, используя блокировку чтения-записи. Изучаемые системные вызовы: `fcntl(F_GETLK, F_SETLK, F_SETLKW, F_UNLK)`.

Программа в режиме конкурентного доступа читает из и пишет в файл, содержащий записи фиксированного формата. Формат записей произвольный. Примерный формат записи:

```
struct record_s {
     char name[80]; // Ф.И.О. студента
     char address[80]; // адрес проживания
     uint8_t semester; // семестр
};
```

Файл должен содержать не менее 10 записей. Создается и наполняется с помощью любых средств.

Программа должна выполнять следующие операции:

1) отображение содержимого файла с последовательной нумерацией записей (`LST`);

2) получение записи с порядковым номером `Rec_No (GET Rec_No)`;

3) модификацию полей записи;

4) сохранение последней прочитанной и модифицированной записи по месту (`PUT`).

Интерфейс с пользователем на «вкус» студента.

Алгоритм конкурентного доступ к записи:

```
REC <-- get(Rec_No) // читаем запись в буфер REC
Again:
REC_WRK <-- REC // копируем в рабочую область
... // делаем что-нибудь с записью в REC_WRK и желаем ее сохранить
if (REC_WRK != REC) { //
lock(Rec_No) // блокируем запись для модификации в файле
REC_NEW <-- get(Rec_No) // и перечитываем запись из файла
if (REC_NEW != REC) { // кто-то изменил запись после ее получения
unlock(Rec_No) // освобождаем запись
message // сообщаем, что запись кто-то изменил
REC <-- REC_NEW // повторим все с ее новым содержимым
goto Again
}
put(REC_WRK, Rec_No) // сохраняем новое содержимое
unlock(Rec_No) // освобождаем запись
}
```

Для отладки и тестирования используется не менее двух экземпляров программы.

Для блокирования записей используется `fcntl()` и исключительные блокировки на основе файловых описаний (`F_OFD_*`).

2 ОПИСАНИЕ АЛГОРИТМОВ И РЕШЕНИЙ

2.1 Общая структура

Программа представляет собой интерактивную систему для работы с записями студентов, хранящимися файле. Основные функции:

Чтение и вывод записей (просмотр всех или конкретной записи).

Изменение данных с проверкой на конфликты.

Блокировка записей для обеспечения согласованности при многопользовательской работе.

Автоматический повтор операции при обнаружении изменений другим процессом. Программа спрашивает у пользователя, необходимо ли заменить данные, которые были изменены другим процессом.

Программа использует низкоуровневые файловые операции (`POSIX API`) для работы с данными, включая блокировки на уровне записей (`fcntl`).

2.2 Алгоритм работы основного процесса

Алгоритм работы основного процесса:

1) инициализация;
2) открытие файла данных (создание, если не существует);
3) проверка и инициализация тестовыми данными при пустом файле;
4) главный цикл обработки команд;
5) вывод меню с доступными операциями;
6) ожидание ввода команд:

`LST` - вывод всех записей;

`GET` - чтение конкретной записи;

`MOD` - модификация текущей записи;

`PUT` - сохранение изменений с проверкой на конфликты;

`EXIT` - завершение работы.

7) обработка команд;
8) завершение работы;
9) корректное закрытие файла;
10) освобождение ресурсов.

3 ФУНКЦИОНАЛЬНАЯ СТРУКТУРА ПРОЕКТА

Система состоит из нескольких ключевых модулей, каждый из которых выполняет определенную функцию.

Модуль управления файлом записей (`file_operations`) выполняет следующие функции:

1) Инициализация файла с тестовыми записями при первом запуске;

2) Реализация механизма блокировок (`F_RDLCK, F_WRLCK`) для безопасного многопользовательского доступа. Автоматическое снятие блокировок при завершении операций;

3) `get_record()` - атомарное чтение записи с проверкой блокировок;

4) `put_record()` - атомарная запись с fsync для гарантии сохранности;

5) `get_total_records()` - определение количества записей в файле;

6) `print_record()` - форматированный вывод одной записи;

7) `list_all_records()` - безопасный вывод всех записей с поочередной блокировкой.

Модуль пользовательского интерфейса (`user_interface`) выполняет следующие функции:

1) `show_menu()` - отображение доступных операций;

2) `get_user_choice()` - обработка ввода пользователя;

3) `run_main_loop()` - основной цикл обработки команд;

4) `handle_list_command()` - вывод всех записей;

5) `handle_get_command()` - загрузка конкретной записи с проверкой несохраненных изменений;

6) `handle_modify_command()` - редактирование полей записи;

7) `handle_put_command()` - сохранение с обнаружением конфликтов;

8) `clear_input()` - очистка буфера ввода.

Модуль управления программой (`main`) выполняет следующие функции:

1) Инициализация системы. Обработка аргументов командной строки. Открытие/создание файла данных. Установка обработчика сигналов;

2) Запуск пользовательского интерфейса.

Модуль структуры данных (`data_structures`) включает следующее:

1) Дескриптор файла `fd`;

2) Структура для блокировок `fl`;

3) Константы (`MAX_ADDRESS_LEN, MAX_NAME_LEN`);

4) `record_t` - структура записи (имя, адрес, семестр);

5) Типы блокировок (`REC_LOCK_READ`, `REC_LOCK_WRITE`).

4 ПОРЯДОК СБОРКИ И ЗАПУСКА ПРОЕКТА

Порядок сборки и запуска состоит в следующем:
1) Разархивировать каталог с проектом;
2) Перейти в каталог с проектом
`$cd "Слепица О.Н./lab07";`
3) Собрать проект используя `make`;
4) После сборки проекта можно использовать, прописав
`$./build/release/main`

5 РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

```
helga@fedora:~/tar_working_dir/Слепица О.Н./lab07$
./build/release/main
=== Лабораторная работа №7: Блокировки чтения/записи ===
Программа для работы с записями студентов
Файл данных успешно открыт: students.dat
Доступно записей: 10
=== МЕНЮ ===
1. LST  - Показать все записи
2. GET  - Получить запись по номеру
3. MOD  - Модифицировать текущую запись
4. PUT  - Сохранить изменения
5. EXIT - Выход
============
Выберите команду: 2
Введите номер записи: 1
Получение записи #1...
=== Запись #1 ===
Ф.И.О.:    Петр
Адрес:     СПб
Семестр:   2
==================
Запись #1 загружена в буфер.
=== МЕНЮ ===
1. LST  - Показать все записи
2. GET  - Получить запись по номеру
3. MOD  - Модифицировать текущую запись
4. PUT  - Сохранить изменения
5. EXIT - Выход
============
Текущая запись: #1
Выберите команду: 3
=== Модификация записи #1 ===
Текущие данные:
=== Запись #1 ===
Ф.И.О.:    Петр
Адрес:     СПб
Семестр:   2
==================
Введите новые данные (Enter - оставить без изменений):
Ф.И.О. [Петр]: Саша
Адрес [СПб]:
Семестр [2]:
Запись изменена в буфере. Используйте PUT для сохранения.
Новые данные:
=== Запись #1 ===
Ф.И.О.:    Саша
Адрес:     СПб
```

```
Семестр:   2
==================
=== МЕНЮ ===
1. LST  - Показать все записи
2. GET  - Получить запись по номеру
3. MOD  - Модифицировать текущую запись
4. PUT  - Сохранить изменения
5. EXIT - Выход
============
Текущая запись: #1 (изменена)
Выберите команду: 4
Сохранение записи #1...
ВНИМАНИЕ: Запись была изменена другим процессом!
Версия при загрузке:
=== Запись #1 ===
Ф.И.О.:    Петр
Адрес:     СПб
Семестр:   2
==================
Текущие данные в файле:
=== Запись #1 ===
Ф.И.О.:    Ольга
Адрес:     Минск
Семестр:   2
==================
Ваши изменения:
=== Запись #1 ===
Ф.И.О.:    Саша
Адрес:     СПб
Семестр:   2
==================
Выберите действие:
1. Перезаписать изменения другого процесса
2. Отменить свои изменения
3. Попробовать снова (повторить чтение)
Ваш выбор: 2
Изменения отменены. Загружена актуальная версия.
=== МЕНЮ ===
1. LST  - Показать все записи
2. GET  - Получить запись по номеру
3. MOD  - Модифицировать текущую запись
4. PUT  - Сохранить изменения
5. EXIT - Выход
============
Текущая запись: #1
Выберите команду: 5
Завершение работы...
Программа завершена.
```