МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №0
по курсу «Операционные системы»

Выполнил: И. О. Фамилия
Группа: М8О-ХХХБ-ХХ
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Цель работы

Изучение механизмов создания процессов, организации межпроцессного взаимодействия через pipes и обработки данных в многопроцессной архитектуре.

Задание

Правило фильтрации: строки длины больше 10 отправляются в pipe2, четные в pipe1. Дочерние процессы удаляют все гласные из строк.

Вариант

17

Метод решения

Данная программа реализует многопроцессную обработку текстовых данных с использованием каналов (pipes) для межпроцессного взаимодействия. Основной алгоритм: родительский процесс читает строки из стандартного ввода и направляет строки длиной больше 10 второму дочернему процессу, меньше 10 - первому. Каждый дочерний процесс получает строки из своего канала, удаляет все гласныеиз строк их и записывает в указанный файл.

Ключевые компоненты:
ParentProcess - управляет каналами и дочерними процессами
Pipe - реализация каналов
ChildProcess - запускает дочерние процессы
ChildProcessor - обрабатывает данные в дочерних процессах

Системные вызовы:
Linux: pipe, fork, execl, read, write

Программа использует объектно-ориентированный подход с инкапсуляцией.

Описание программы

Программа реализует многопроцессную обработку текстовых данных через каналы (pipes). Родительский процесс читает строки из стандартного ввода и распределяет их между двумя дочерними процессами

Каждый дочерний процесс удаляет все гласные буквы и записывает результат в указанный файл.

Программа состоит из двух исполняемых файлов: `parent` и `child`.
`parent` (`src/parent.cpp`) — родительский процесс:
создаёт каналы (`Pipe`), порождает дочерние процессы (`fork/execl`),
считывает строки до `END` и распределяет их по длине.
`child` (`src/child.cpp`) — дочерний процесс:
через `ChildProcess` читает данные из канала,
удаляет гласные с помощью `ChildProcessor`
и записывает результат в файл и на экран.

Результаты

Разработанная программа успешно реализует многопроцессную архитектуру для параллельной обработки текстовых данных.
В ходе решения были достигнуты следующие ключевые результаты:
Корректная работа системы межпроцессного взаимодействия
Реализованы два независимых канала передачи данных между родительским и дочерними процессами
Обеспечено четкое распределение строк по принципу больше/меньше 10
Достигнута синхронизация процессов через блокирующие операции чтения/записи
Реализована унифицированная абстракция для работы с каналами через класс Pipe

Выводы

В ходе лабораторной работы успешно разработана многопроцессная система обработки текстовых данных с использованием межпроцессного взаимодействия через каналы. Программа демонстрирует корректную работу на Unix системах.

Исходная программа

Системные вызовы

Ваш текст о системных вызовах здесь...