**Московский авиационный институт (национальный исследовательский университет)**

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

**Лабораторная работа по предмету "Операционные системы" №1**

Студент: Мартынов С.М.

Преподаватель: Миронов Е.С.

Группа: М8О-201БВ-24

Оценка:\_\_\_\_\_\_\_\_\_

Дата:\_\_\_\_\_\_\_\_\_

Подпись:\_\_\_\_\_\_\_\_\_

**Оглавление**

1. Цель работы
2. Постановка задачи
3. Общие сведения о программе
4. Общий алгоритм решения
5. Реализация
6. Пример работы
7. Вывод

# 

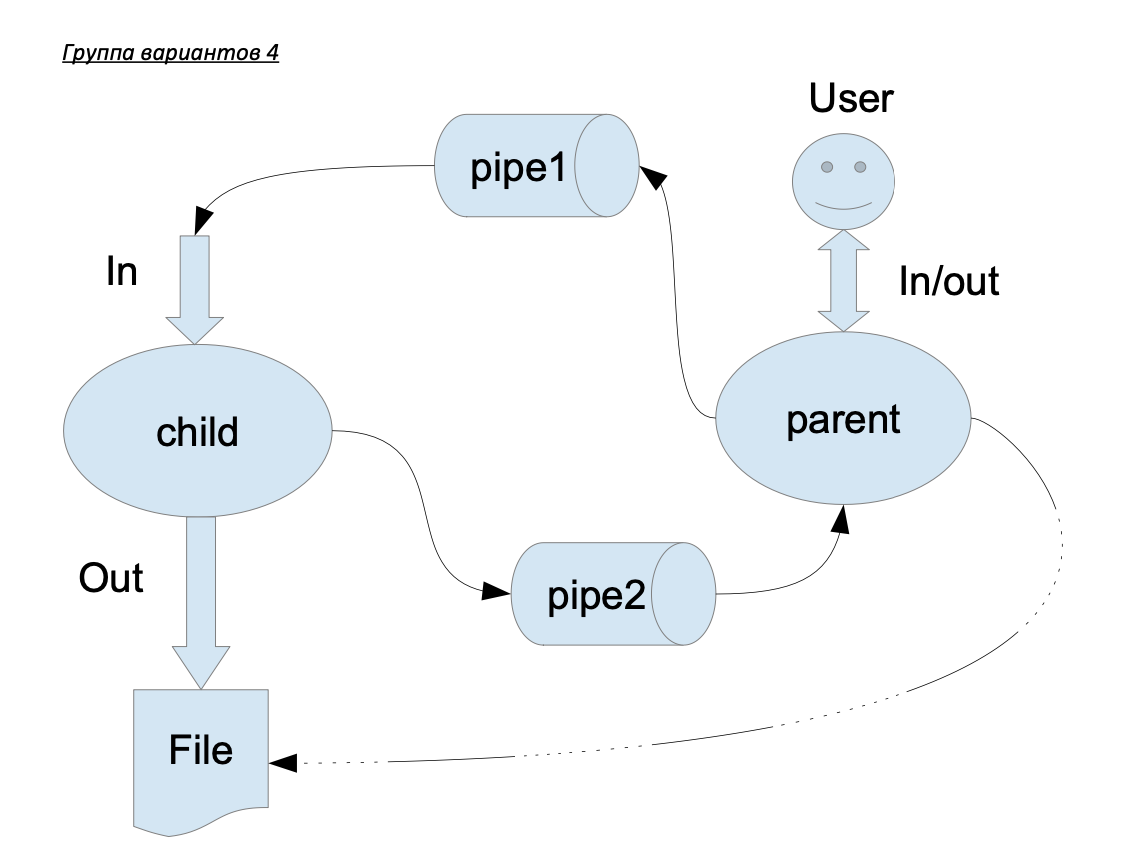
# **Цель работы**

Приобретение практических навыков в:

* Управлении процессами в ОС
* Обеспечении обмена данных между процессами посредством каналов

# **Постановка задачи**

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решения задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

**

Родительский процесс создает дочерний процесс. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child проверяет строки на валидность правилу. Если строка соответствует правилу, то она выводится в стандартный поток вывода дочернего процесса, иначе в pipe2 выводится информация об ошибке. Родительский процесс полученные от child ошибки выводит в стандартный поток вывода.

Вариант 15) Правило проверки: строка должна начинаться с заглавной буквы

# **Общие сведения о программе**

Программа представлена двумя файлами **–** parent.c и child.c

В программе используются следующие системные вызовы:

* pid\_t fork() - создание дочернего процесса
* int execl (const char \*Path, const char \*Argument0, const char \*Argument1, ... , NULL) - замена образа памяти процесса
* pid\_t wait(int \*stat\_loc) - ожидание завершения дочернего процесса
* void exit(int status) - завершения выполнения процесса и возвращение статуса
* int pipe(int pipefd[2]) - создание неименованного канала для передачи данных между процессами
* int dup2(int oldfd, int newfd) - переназначение файлового дескриптора
* int open(const char \*pathname, int flags, mode\_t mode) - открытие\создание файла
* int close(int fd) - закрыть файл

# 

# **Общий алгоритм решения**

1. Создаются два pipe для обмена данными между процессами:
   * pipe1: родитель → дочерний (передача строк)
   * pipe2: дочерний → родитель (передача ошибок)
2. Родительский процесс вызывает fork()

В дочернем процессе:

* + Stdout перенаправляется на запись в файл
  + Stderr перенаправляется на запись в pipe2
  + Stdin перенаправляется на чтение из pipe1

В родительском процессе:

* Stdin остается стандартным
* Stdout остается стандартным

1. Дочерний процесс:  
   * Получает имя файла из pipe1 (первая строка)
   * Создаёт файловый дескриптор для файла
   * Построчно читает строки из pipe1
   * Проверяет каждую строку по правилу: начинается с заглавной буквы
   * Если строка валидна - записывает ее в файл (выводим в stdout)
   * Если невалидна - отправляет сообщение об ошибке в pipe2
2. Родительский процесс:  
   * Получает от пользователя имя файла и отправляет его в pipe1
   * В цикле получает строки от пользователя и отправляет в pipe1
   * Получает сообщения об ошибках из pipe2 и выводит на экран
   * Завершает работу после получения всех сообщений об ошибках
3. Родитель ждёт завершения дочернего процесса и завершает работу

**Реализация  
  
Git:** [**мой гит**](https://github.com/SergioMartinov31/OS)

# 

# **Пример работы**

|  |  |
| --- | --- |
| Input | Out |
| hello world  TEST  123Start  Apple | TEST  Apple |
| !Start  $Money  45  First | First |
|  |  |

# **Вывод**

В ходе выполнения лабораторной работы я освоил не только базовые механизмы работы с процессами и межпроцессного взаимодействия через pipe, но и выделил несколько важных практических аспектов.

Первый ключевой момент - ограничение, с которым я столкнулся: проблема невозможности реализовать пока последовательный вывод ошибок из-за блокирующего чтения. Невозможность мгновенно получать отчёт о валидности строки наглядно демонстрирует, почему в реальности часто требуется многопоточность, где один поток отвечает за взаимодействие с пользователем, а другой — за обработку данных, в нашем случае ошибок из pipe2.

Второй важный вывод касается международной поддержки. Я обнаружил, что стандартная функция isupper() из библиотеки ctype.h работает только с латинскими символами ASCII и не поддерживает кириллицу.