**Отчет по лабораторной работе №1** по курсу Операционные системы

Студент группы М8О-201БВ-24 Рахматуллин Айдар Рамильевич, № по списку 27

Контакты e-mail: proo100top@mail.ru

Работа выполнена: «20» сентября 2025 г.

Преподаватель: \_\_\_\_\_\_\_ каф. 806 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

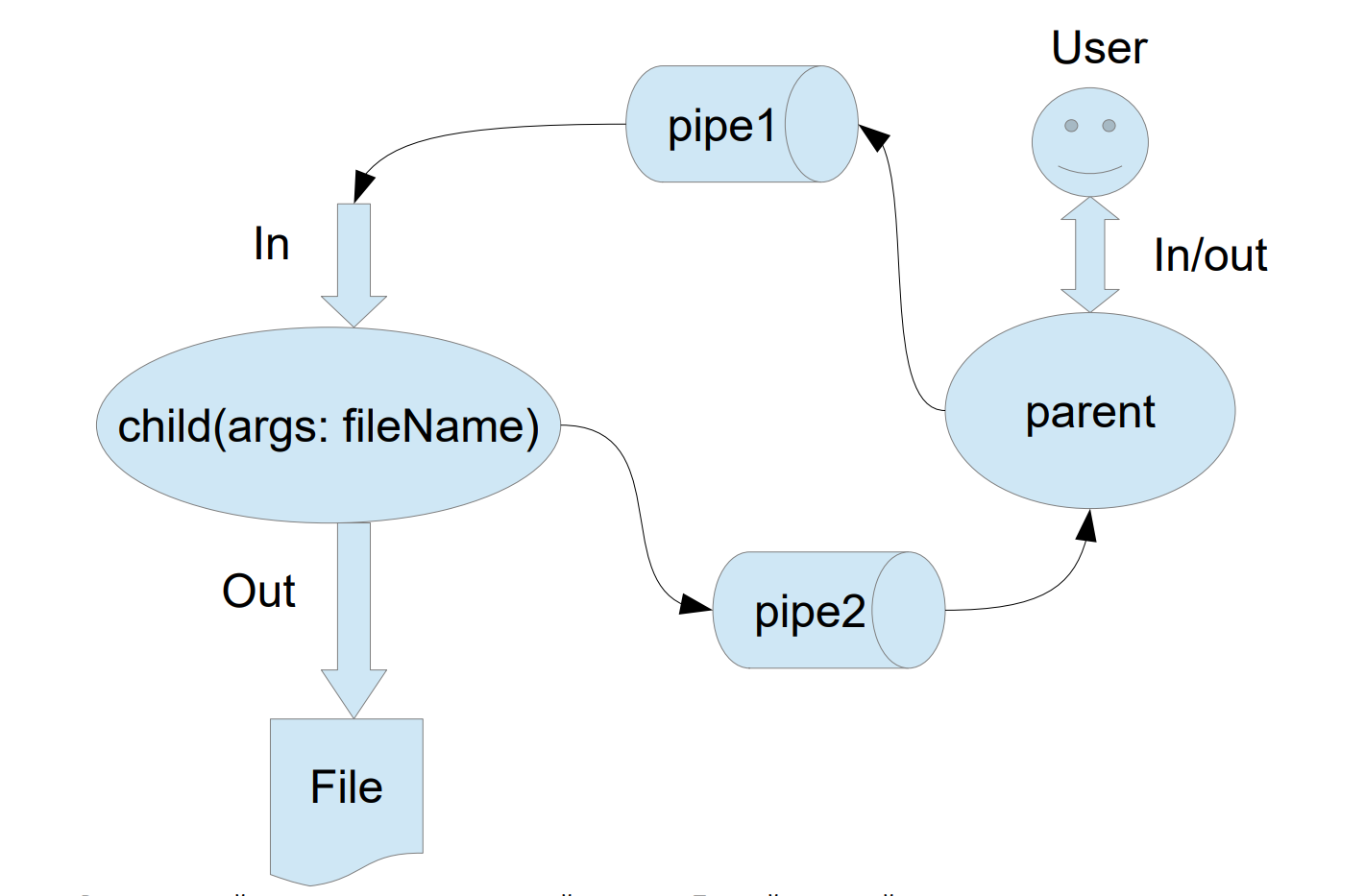
Входной контроль знаний с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчет сдан « » \_\_\_\_\_\_\_\_\_202 \_\_ г., итоговая оценка \_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. **Тема:** Процессы в ОС.

1. **Цель работы:** Приобретение практических навыков в: управление процессами в ОС, обеспечение обмена данных между процессами посредством каналов
2. **Задание** (*вариант №* **1**)**:**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 решение 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 строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс при необходимости передает данные в родительский процесс через pipe2. Результаты своей работы дочерний процесс пишет в созданный им файл. Допускается просто открыть файл и писать туда, не перенаправляя стандартный поток вывода.

1 вариант) Пользователь вводит команды вида: «число число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип int. Количество чисел может быть произвольным.

1. **Оборудование** (лабораторное):

ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ , процессор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, имя узла сети \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ с ОП \_\_\_\_\_\_\_\_\_\_ Мб, НМД \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мб. Терминал \_\_\_\_\_ адрес \_\_\_\_\_\_\_\_\_\_. Принтер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Другие устройства \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Оборудование ПЭВМ студента, если использовалось:*

Процессор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ с ОП \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ГБ, НМД \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ГБ. Монитор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Другие устройства \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Программное обеспечение (лабораторное):**

Операционная система семейства \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, наименование \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

интерпретатор команд \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Система программирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Редактор текстов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Утилиты операционной системы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Прикладные системы и программы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Местонахождение и имена файлов программ и данных \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Программное обеспечение ЭВМ студента, если использовалось:*

Операционная система семейства \_\_\_\_\_\_\_\_\_\_, наименование \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

интерпретатор команд \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Система программирования \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Редактор текстов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ версия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Утилиты операционной системы, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Прикладные системы и программы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Местонахождение и имена файлов программ и данных на домашнем компьютере \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**6. Идея, метод, алгоритм** решения задачи(в формах:словесной,псевдокода,графической[блок-схема,диаграмма,рисунок,таблица] или формальные спецификации с пред- и постусловиями)

**Идея**:

От родительского процесса создать, с помощью fork(), дочерний процесс. Создать pipe, для передачи данных, между родительским процессом (parent.c) и дочерним (child.c). Перенаправить необходимые потоки, а именно стандартный ввод (stdin) дочернего процесса переопределим на файл, а выход (stdout) на pipe. В родительском процессе читаем данные из pipe.

**Метод**:

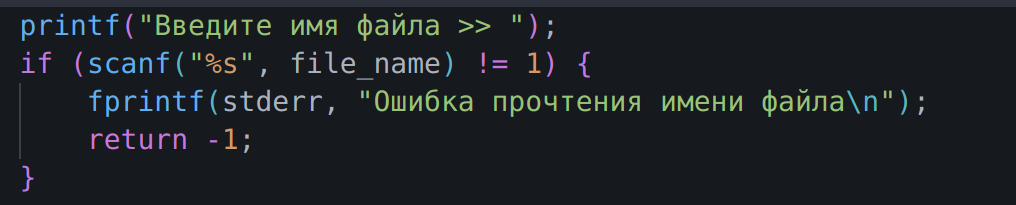
1. Взаимодействие между родительским и дочерним процессом реализуем через pipe.
2. Перенаправление стандартных потоков через dup2.
3. Разделение процессов реализуем, через execl:

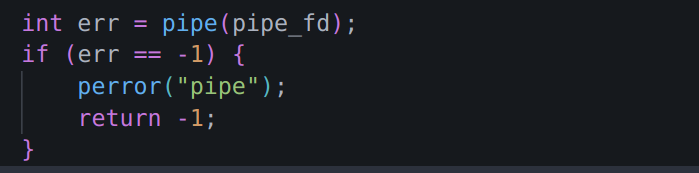
* Дочерний процесс: чтение и обработка данных.
* Родительский процесс: управление и вывод данных.

**Алгоритм**:

parent.c:

При запуске программы parent.c создается процесс - родительский. В этом родительском процессе в начале программы считывается имя файла, с числами вещественного типа. Количество чисел по условию не известно.

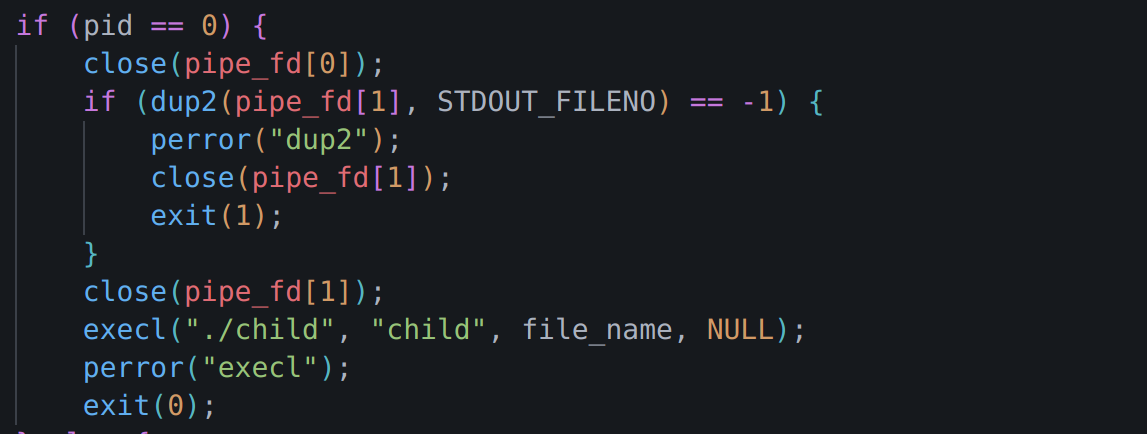
 Затем создается pipe, своего рода очередь, работающая по принципу FIFO. Функция pipe() создает два дескриптора, один pipe\_fd[0] - на чтение, второй pipe\_fd[1] - на запись. При успехе pipe возвращает значение 0, при неудаче -1. После создания pipe происходит обработка ошибки.



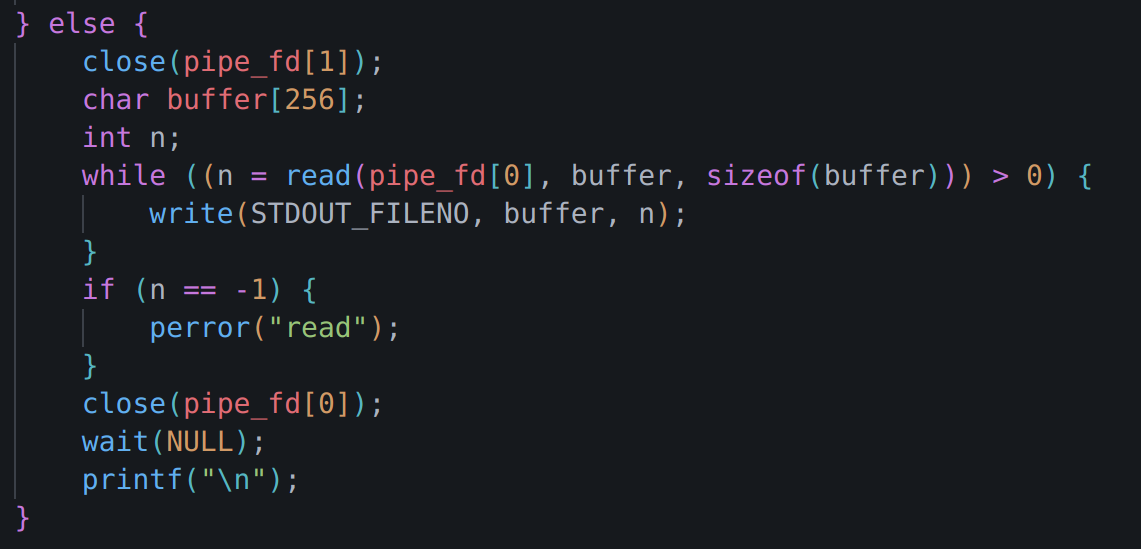
Следом происходит создание дочернего процесса, с помощью функции fork которая возвращает pid процесса типа pid\_t.



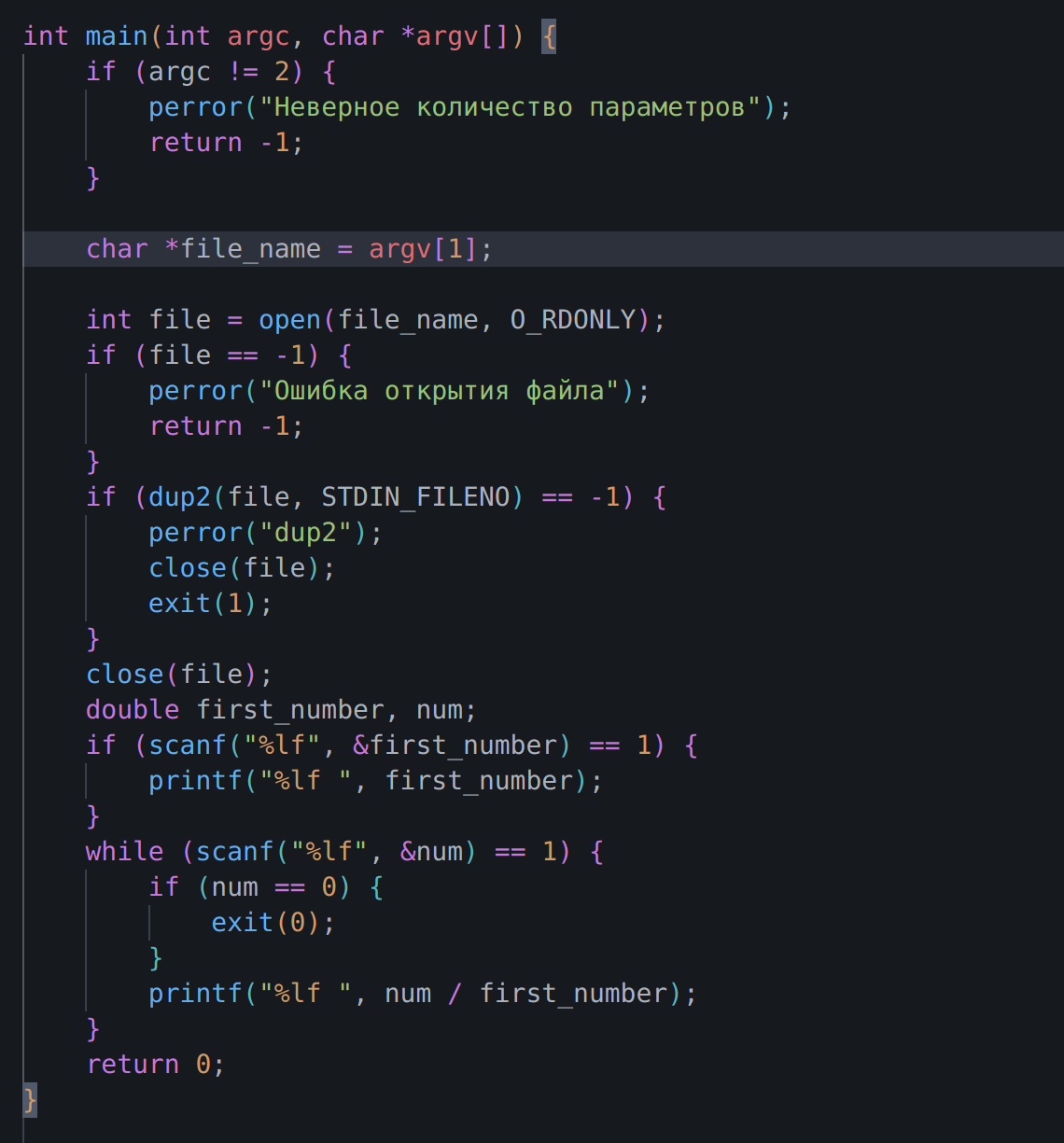
Если процесс является дочерним то его pid в теле родительской программы считается равной 0. Благодаря этому, в родительском процессе можно разделить логику процессов, с помощью условных операторов. Если процесс дочерний (pid == 0) закрываем дескриптор на чтение, т.к. дочерний процесс не будет читать данных из pipe, а только писать в него. Создаем копию дескриптора pipe\_fd[1] и присваиваем этой копии значение STDOUT\_FILENO, другими словами перенаправляем стандартный выход stdout child в pipe. Закрываем старый дескриптор. Подменяем дочерний процесс программой child.c (с тем же pid), с помощью execl, в аргументы которой передаем путь к файлу child.c и имя файла с числами. Производим проверку execl, если execl возвращается с ошибкой завершаем процесс exit(0).



Если процесс родительский (т.е. его pid != 0 в теле своей программы), закрываем дескриптор на запись, т.к. родительский процесс не будет писать в pipe, а только читать из него. Читаем кусками по 256 байт из pipe данные, пока они есть, и сразу пишем их в stdout родительского процесса т.е. в консоль. При возникновении ошибок, сообщаем о возникновении ошибки. Закрываем pipe и дожидаемся завершения дочернего процесса.



child.c:



В функции main принимаем аргумент функции execl в виде имени файла. Открываем этот файл только на чтение (O\_RDONLY - read only) и получаем файловый дескриптор file. Создаем копию файлового дескриптора file и присваиваем этой копии значение STDIN\_FILENO и закрываем старый дескриптор file, другими словами stdin дочернего процесса теперь является file, при чтении из stdin мы сразу читаем из file. Читаем сначало одно число - делитель остальных чисел, потом читаем оставшиеся числа, делим их на первое и отправляем их в stdout т.е. в pipe.

При чтении чисел проверяем не равны ли они нулю. Если равны завершаем процесс.

### 

*Пункты 1-6 отчета составляются строго до начала лабораторной работы.*

*Допущен к выполнению работы.*

**Подпись преподавателя**

**7. Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами,подписанныйпреподавателем)

**8. Дневник отладки** должен содержать дату и время сеансов отладки, и основные события(ошибки в сценарии и программе,нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Лаб. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
|  |  |  |  |  |  |  |

**9.Замечания автора** по существу работы:

**10. Выводы**

В ходе выполнения лабораторной работы я получил важные практические навыки, которые значительно расширили моё понимание работы операционной системы и принципов программирования. Эта работа позволила мне узнать что такое процессы, как ими управлять, как организовать межпроцессное взаимодействие.

Я научился создавать новые процессы с помощью функции fork(). Понял, как различать родительский и дочерний процессы по их PID. Узнал о pipe и научился использовать его для передачи данных между процессами. Полученные знания и навыки могут быть применены для разработки более сложных программ.