МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем

и технологий

**Отчет**

**по лабораторной работе № 10**

по дисциплине: ”Системное программирование”

на тему: ***”*РАБОТА С КАНАЛАМИ (Pipes)*”***

Выполнил**:** студент группы *10702121* Ахраменко Н. В.

Козлов М. А.

Василевский А. А.

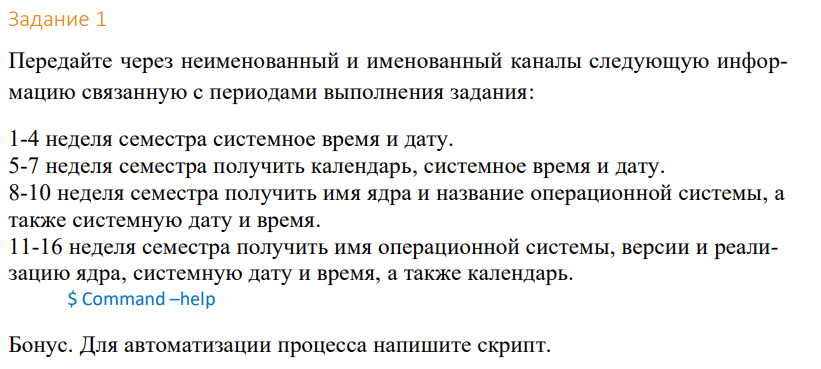
Коржицкий Д. В.

Принял**:** пр. Мисякова В. А.

Минск 2023

# Лабораторная работа №10.

**Цель работы:** Освоить механизм взаимодействия между процессами на основе каналов (pipes).

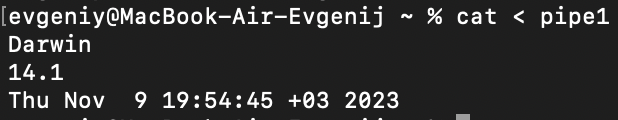
  
**Решение**

**Неименованный канал:**

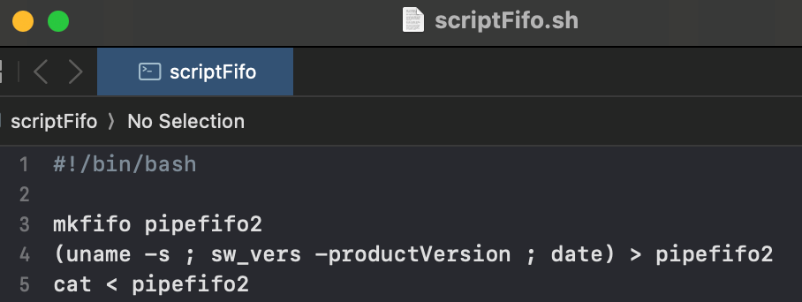
**Создаем обычный файл и перенаправляем вывод в него.**

****

**Получаем информацию из неименованного(pipe) канала.**

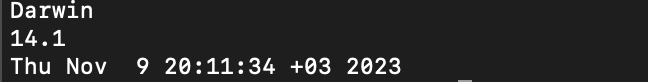


**Пишем скрипт для автоматизации процесса**

****

**Запускаем скрипт**

****



**Именованный канал:**

**Создадим специальный файл-канал(тип "pipe") с помощью команды mkfifo.**

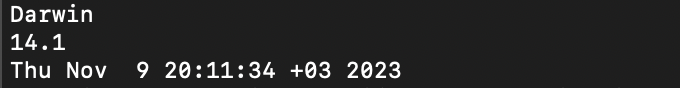
****

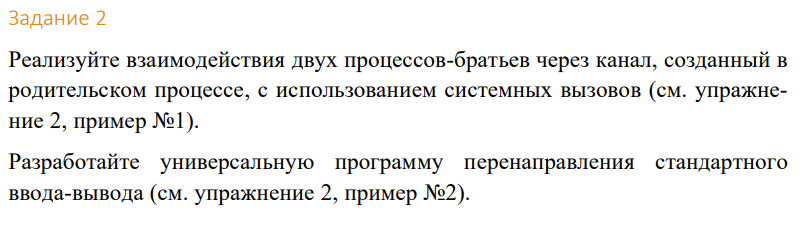
**Передаем данные в файл канала(имя ядра, название ОС, дата и время)**

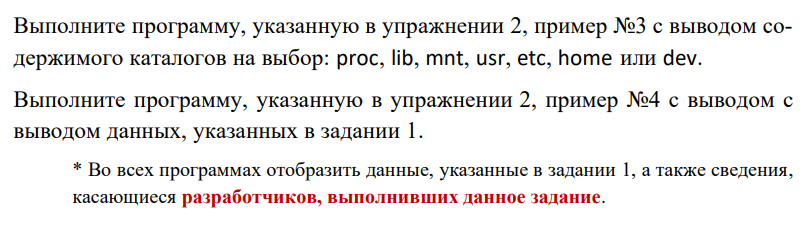
****

**Получаем информацию из именованного(fifo) канала**

****

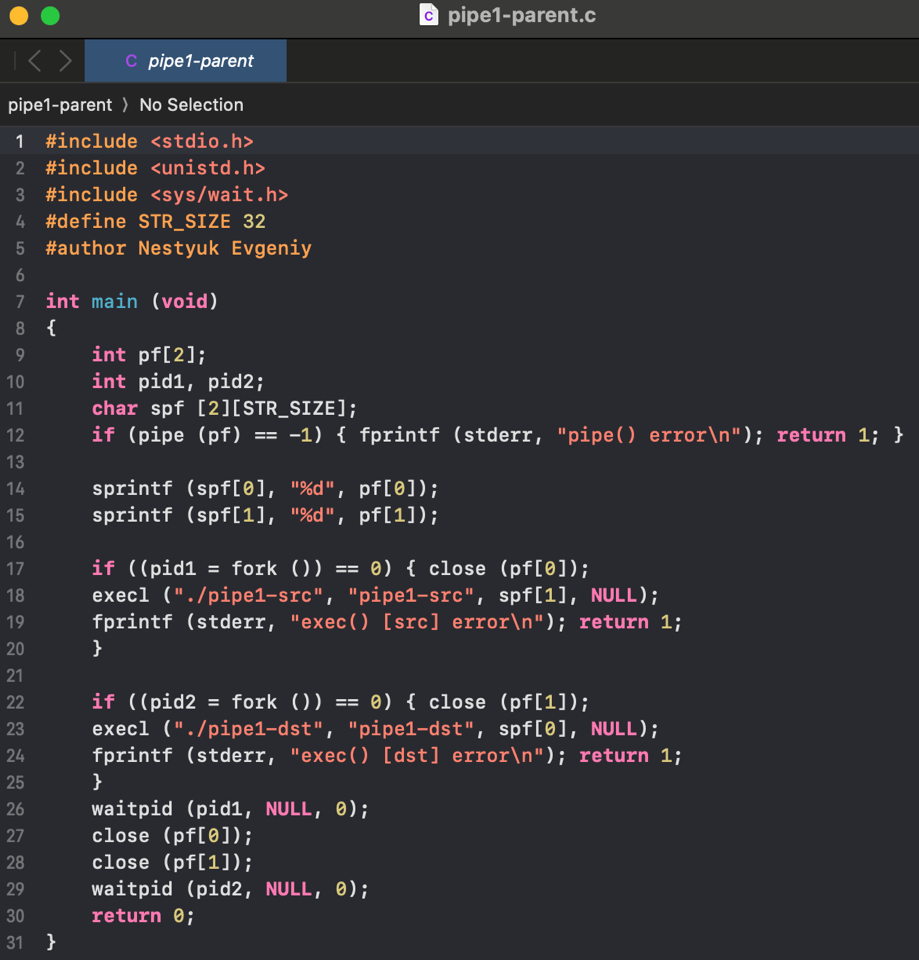
****

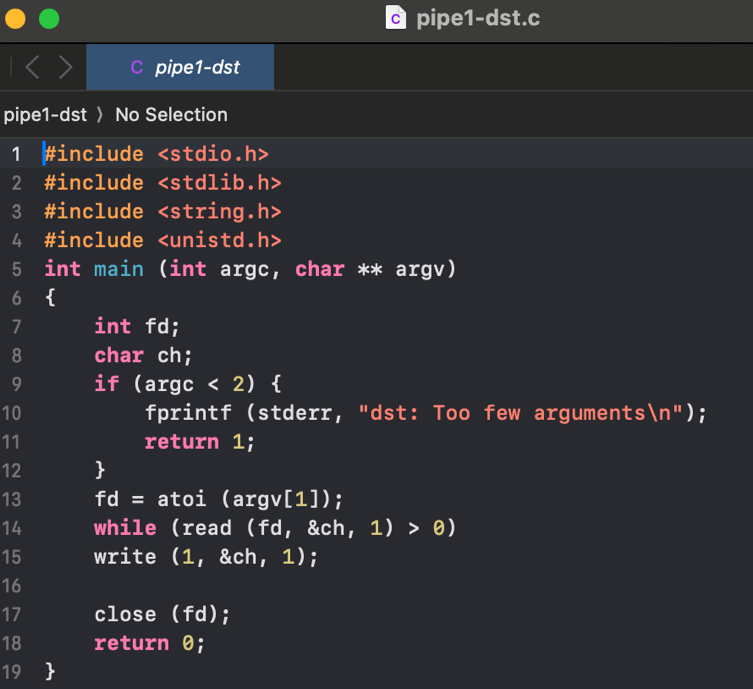


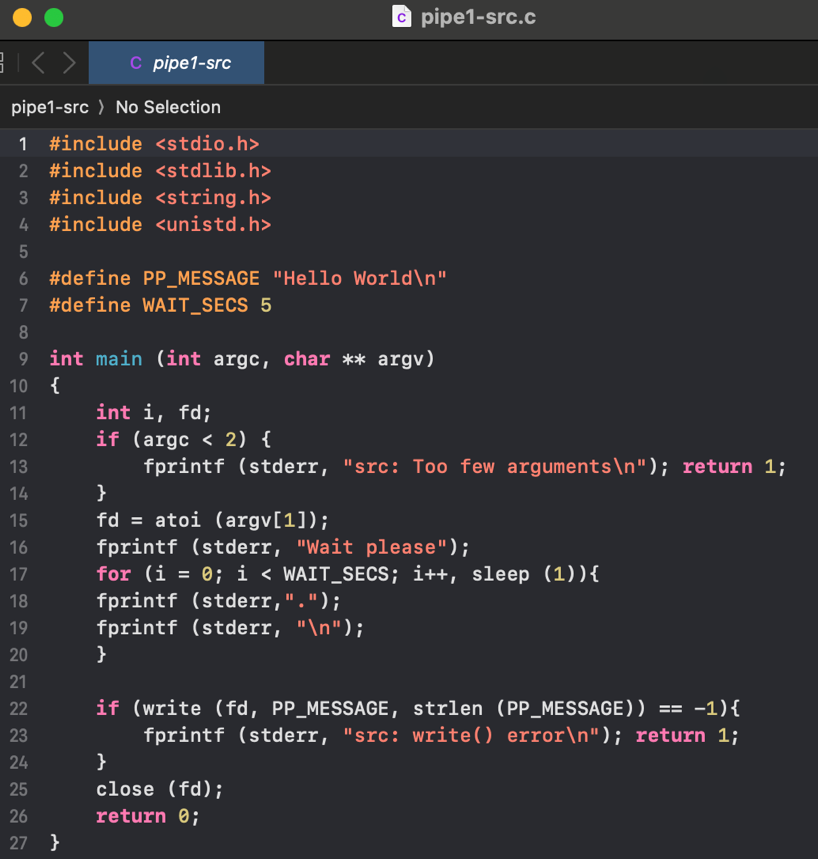


**Решение. Реализуем взаимодействие двух процессов-братьев через канал, созданный в родительском процессе, с использованием системных вызовов**

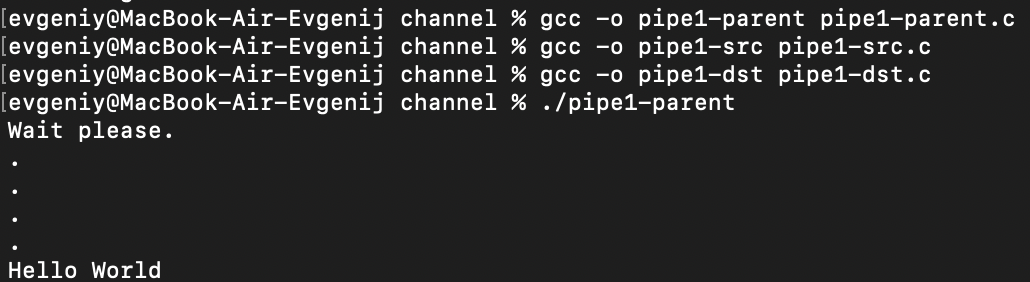
**Содержимое файлов:**





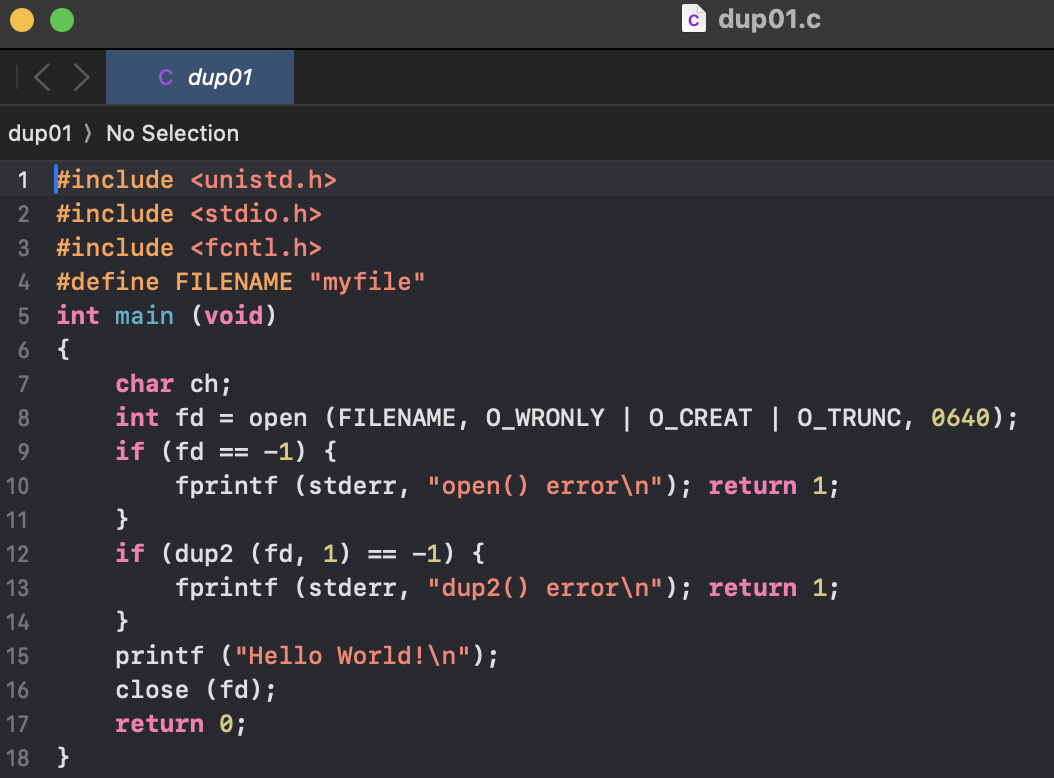
****

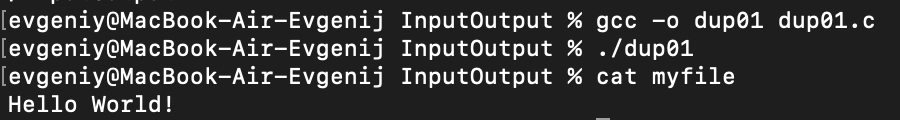
**Произведем автосборку многофайлового проекта.**



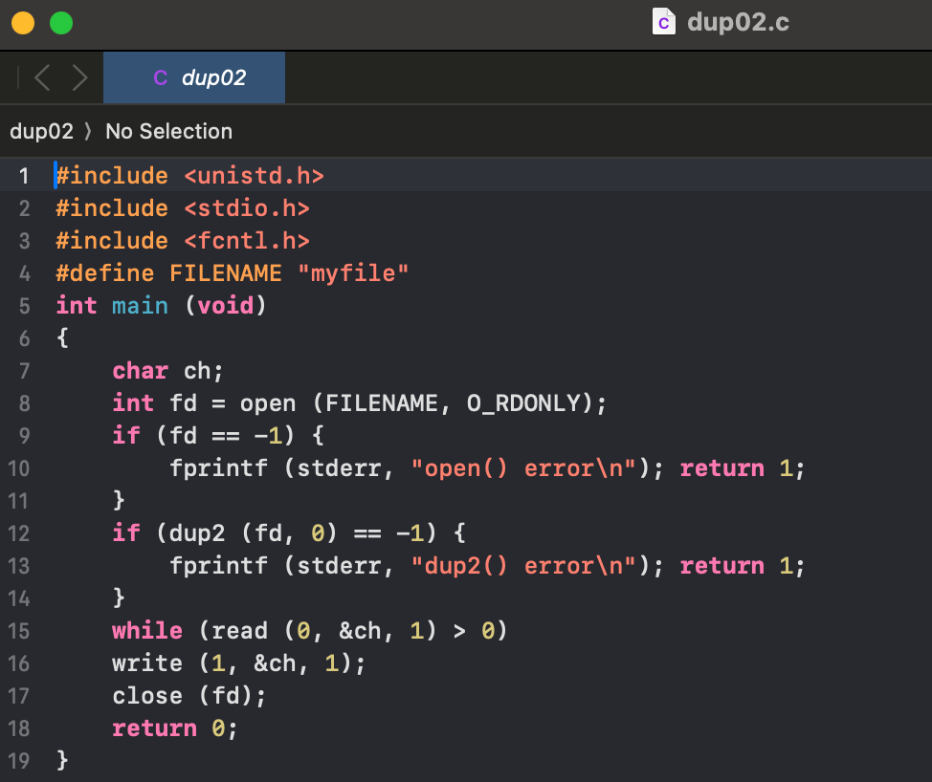
**Решение2. Разработаем универсальную программу перенаправления стандартного ввода-вывода.**

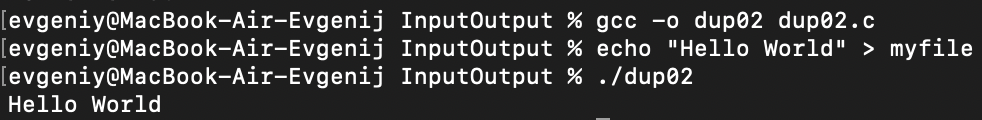
**Перенаправление стандартного вывода в файл.**





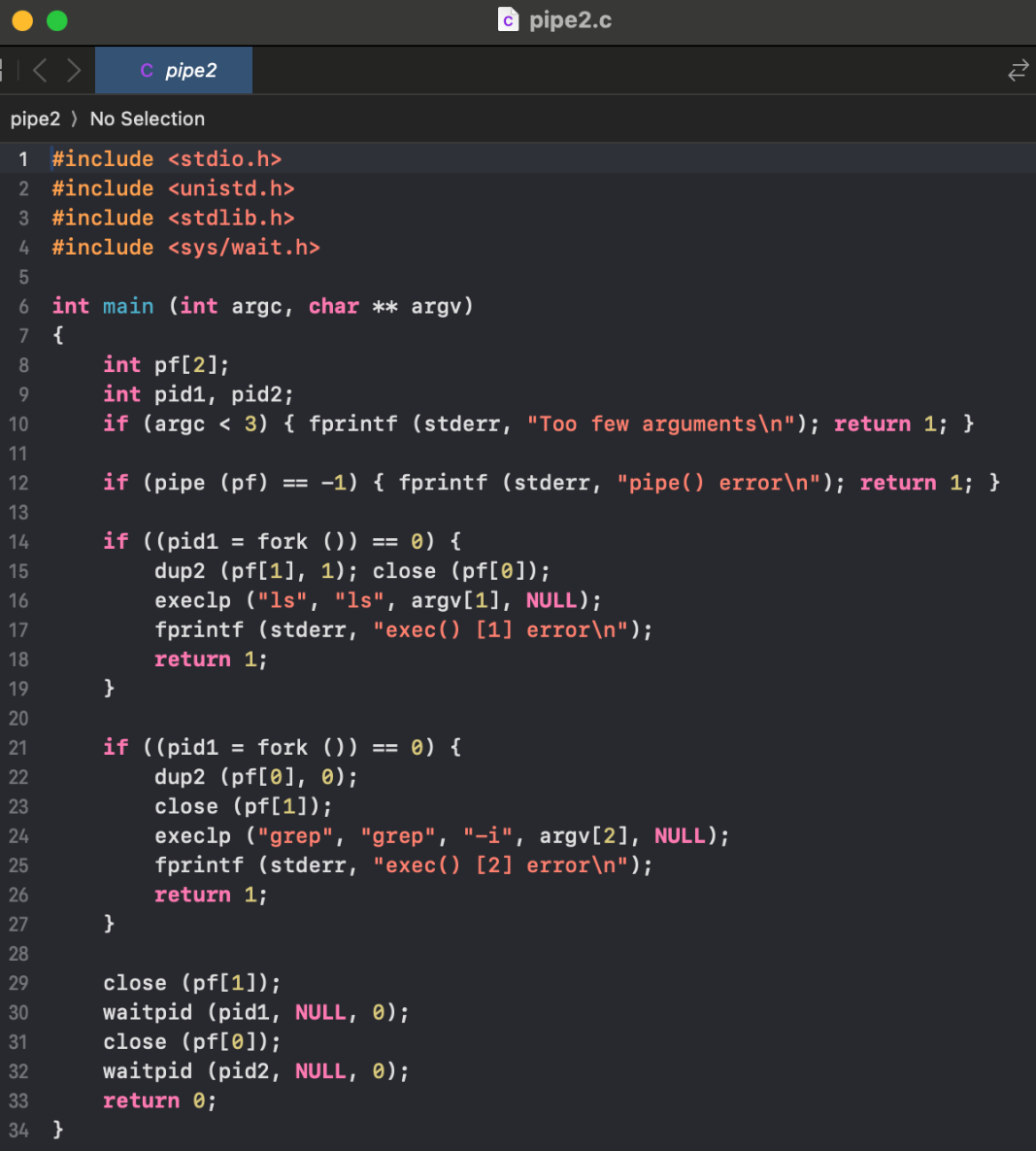
**Перенаправление стандартного ввода.**



****

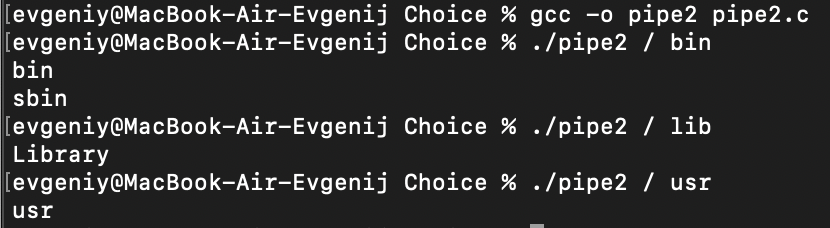
**Решение 3. Вывод содержимого каталог на выбор**

**Содержимое файла**

****

**Сборка**

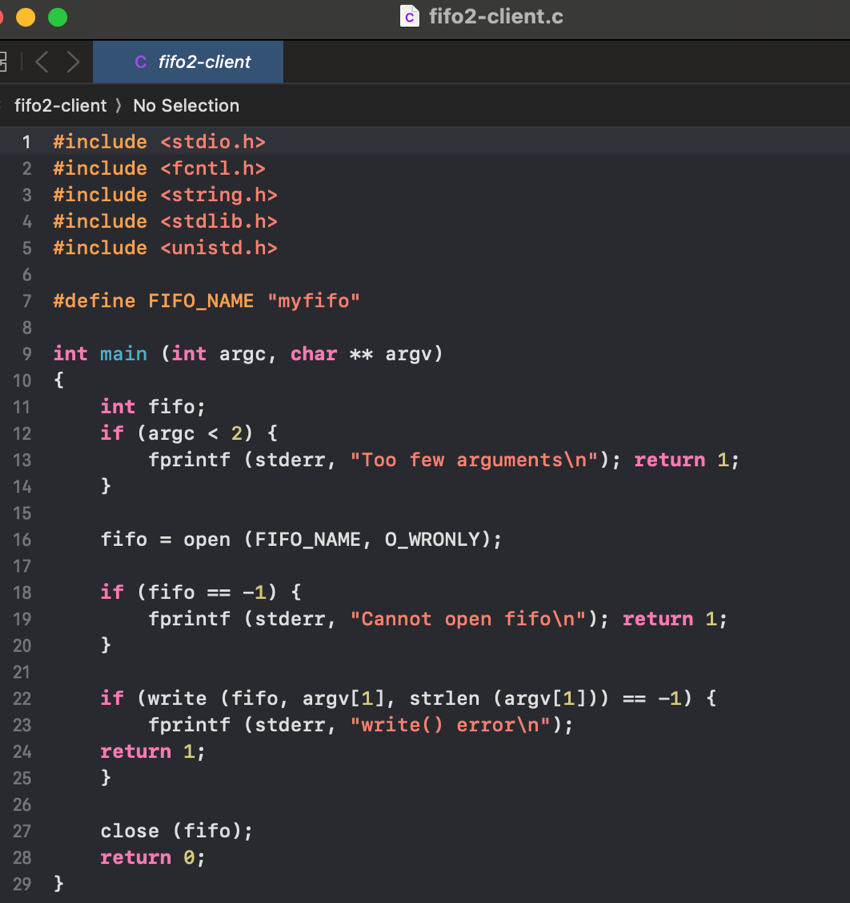
****

****

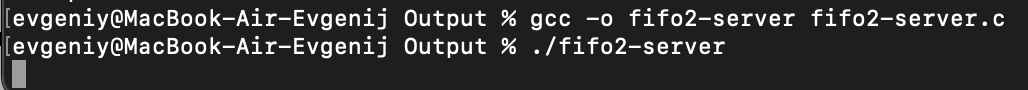
**Решение 4. Вывод данных**

**Содержимое файлов:**

****

****

**Запустим "сервер", он создаст FIFO и "замрет" в ожидании поступления данных:**

****

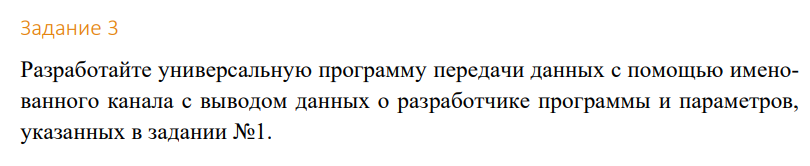
**Запустим в другом терминальном окне клиентскую программу:**

****

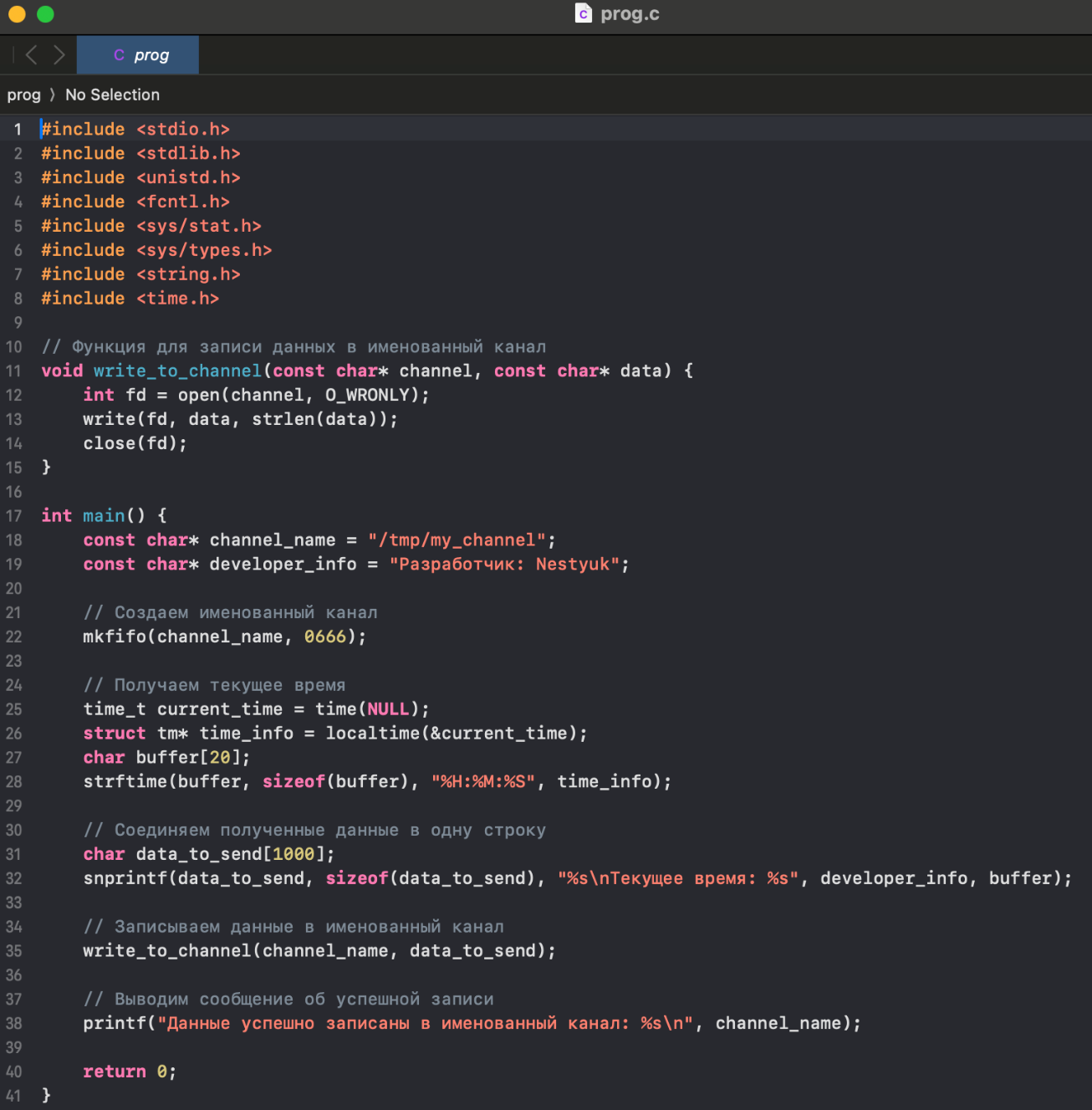
****

**Серверный процесс тут же "проснется", выведет сообщение "Nestyuk" и закончит работу**

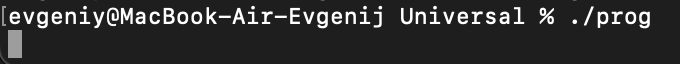
****



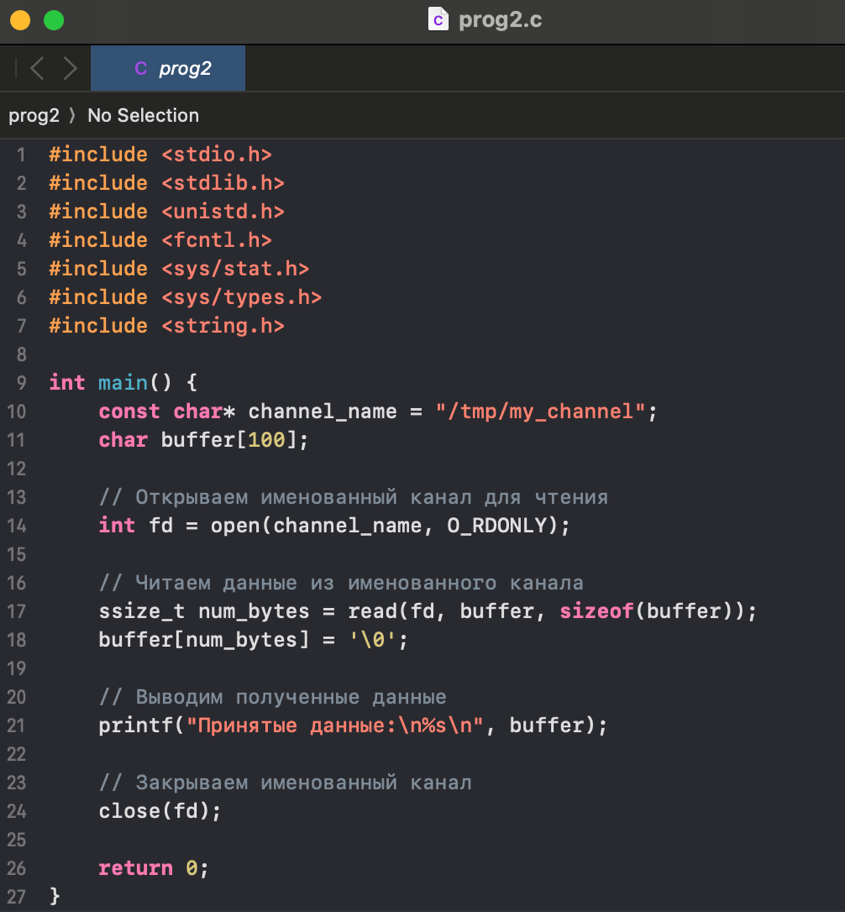
**Содержимое клиента:**

****

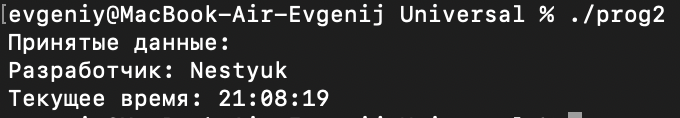
**Запуск клиента:**

****

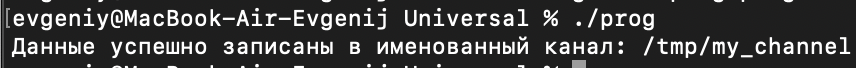
**Содержимое сервера:**

****

**Запуск сервера:**

****

**И закрытие клиента:**



**Контрольные вопросы:**

1) Межпроцессное взаимодействие (IPC) - это механизм обмена данными и синхронизации между различными процессами, которые выполняются на компьютере. IPC необходимо, когда разные процессы должны сотрудничать, обмениваться информацией или координировать свою работу. Это может включать передачу данных между процессами, синхронизацию выполнения задач, обмен сообщениями и другие методы взаимодействия.

2) Каналы (или именованные каналы) - это один из механизмов межпроцессного взаимодействия в операционных системах. Они представляют собой именованные файлы, которые могут быть использованы для обмена данными между процессами. Один процесс может записывать данные в канал, а другой процесс может читать данные из канала. Каналы позволяют процессам общаться и передавать информацию друг другу, даже если они работают в разных контекстах и не имеют общей памяти.

3) Существуют два типа каналов:

**Именованные каналы (FIFO)**: Эти каналы имеют имя в файловой системе и обычно создаются с помощью команды **mkfifo**. Они подходят для обмена данными между процессами, запущенными в разное время или на разных устройствах.

**Неименованные каналы (pipe)**: Эти каналы не имеют имени в файловой системе и обычно создаются с использованием оператора "|". Они предназначены для обмена данными между процессами, запущенными в один момент времени в рамках одной командной строки.3) **Опишите существующие типы библиотек и принципы их использования.**

4) **Именованные каналы (FIFO)**: Именованные каналы (FIFO) - это механизм межпроцессного взаимодействия, который представляет собой специальный тип файлов в файловой системе. Эти файлы имеют имя и могут быть использованы для обмена данными между процессами. Они подходят для обмена данными между процессами, работающими на разных устройствах или в разное время. FIFO работает по принципу очереди: данные, записанные в FIFO, читаются в том порядке, в котором они были записаны.

5) Каналы межпроцессного взаимодействия относятся к виду межпроцессного взаимодействия, который характеризуется следующим образом:

1. По широте охвата:
   * Однородные (или непосредственные) каналы: Эти каналы обеспечивают обмен данными только между двумя конкретными процессами. Они предоставляют прямой канал связи между двумя процессами и не используются для коммуникации между другими процессами.
   * Разнородные (или косвенные) каналы: Эти каналы позволяют процессам обмениваться данными с несколькими другими процессами. Они могут быть использованы для множества взаимодействий между различными процессами.
2. По направлению передачи данных:
   * Двусторонние каналы: Эти каналы позволяют обмену данными между процессами в обоих направлениях. Оба процесса могут отправлять и получать данные.
   * Односторонние каналы: В этом случае обмен данными возможен только в одном направлении, либо от отправителя к получателю, либо наоборот.
3. По характеру доступа:
   * Синхронные каналы: Взаимодействие между процессами с использованием синхронных каналов происходит блокирующим образом. Отправитель блокируется до тех пор, пока получатель не прочтет переданные данные, и наоборот.

Асинхронные каналы: Асинхронные каналы позволяют процессам взаимодействовать без блокировки. Отправитель может продолжать выполнение после отправки данных, и получатель может читать данные, когда это удобно.

6) **Системные вызовы**:

* + **pipe()**: Создает анонимный канал (pipe) для однонаправленного межпроцессного взаимодействия.
  + **mkfifo()**: Создает FIFO (или именованный канал) для межпроцессного взаимодействия.
  + **socketpair()**: Создает пару сокетов для межпроцессного взаимодействия.

1. **Команды**:
   * **mkfifo** (команда): Создает FIFO (или именованный канал) из командной строки.
2. **Библиотечные функции**:
   * **popen()**: Открывает канал для выполнения команды и получения ее вывода или отправки данных в команду.

**fdopen()**: Связывает файловый дескриптор с потоком для чтения или записи

7)

**Команды**:

1. **mkfifo** (команда):

Команда **mkfifo** используется для создания именованных каналов из командной строки.

**Системные вызовы**:

1. **mkfifo()** (системный вызов в C):

Этот системный вызов создает именованный канал (FIFO) в C-программе.

**2. open()** (системный вызов в C):

Используется для открытия именованного канала как файла для чтения и записи в C-программе.

8)

Некоторые из основных атрибутов системных вызовов включают в себя:

* Название системного вызова.
* Типы и количество аргументов, передаваемых системному вызову.
* Возвращаемое значение, которое представляет результат выполнения системного вызова.
* Внутренние коды ошибок, которые могут возникнуть в случае неудачного выполнения системного вызова.
* Права доступа и разрешения на использование системного вызова

9) **Задачи библиотечных функций popen() и pclose()**:

**1. popen()**: Эта библиотечная функция в языке C используется для открытия канала для выполнения внешних команд и получения их вывода или отправки данных в них. Она создает дополнительный процесс (подпроцесс), который выполняет указанную команду, и возвращает указатель на файловый поток (stream), связанный с этим процессом. **popen()** позволяет взаимодействовать с внешней командой, как если бы она была файлом, и читать/писать данные в нее.

**2. pclose()**: Эта библиотечная функция используется для закрытия канала, созданного с помощью **popen()**. Она завершает подпроцесс, связанный с командой, выполняемой **popen()**, и закрывает соответствующий файловый поток. **pclose()** также возвращает статус завершения выполнения команды.

10) Однонаправленное межпроцессное взаимодействие (IPC) - это процесс обмена информацией между процессами, в котором информация передается только в одном направлении, либо от отправителя к получателю, либо наоборот. Примерами однонаправленного IPC могут быть передача данных от процесса-генератора к процессу-консьюмеру через канал, когда генератор создает данные и передает их консьюмеру для обработки.

11) **Архитектура клиент/сервер**: Это модель архитектуры приложений, в которой приложение разделяется на две основные компоненты - клиент и сервер. Клиенты отправляют запросы серверам, и серверы предоставляют услуги или данные клиентам. Это позволяет разделить ответственности между клиентской и серверной сторонами и обеспечить распределенное взаимодействие между компонентами. Эта архитектура часто используется в сетевых приложениях, где клиенты и серверы общаются по сети.

**Архитектура взаимодействия процессов "источник" - "приёмник"**: Это более общая модель, в которой процессы или компоненты обмениваются данными или сообщениями. Один процесс (источник) генерирует данные или сообщения и передает их другому процессу (приемнику). Это взаимодействие может быть однонаправленным или взаимным, и процессы могут быть на одной машине или в разных местах сети. Эта архитектура не ограничивается только клиент-серверными отношениями и может быть применена в разных контекстах, включая локальное межпроцессное взаимодействие или распределенные системы.