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 № 15**

по дисциплине: ”Системное программирование”

на тему: ***”*** **РАЗДЕЛЯЕМАЯ ПАМЯТЬ (Shared memory), СИНХРОНИЗАЦИЯ*”***

Выполнил**:** студент группы *10702121* Нестюк Е.А.

Огризко А.В.

.

Принял**:** пр. Хвитько Е.А.

Минск 2023

# Лабораторная работа №15.

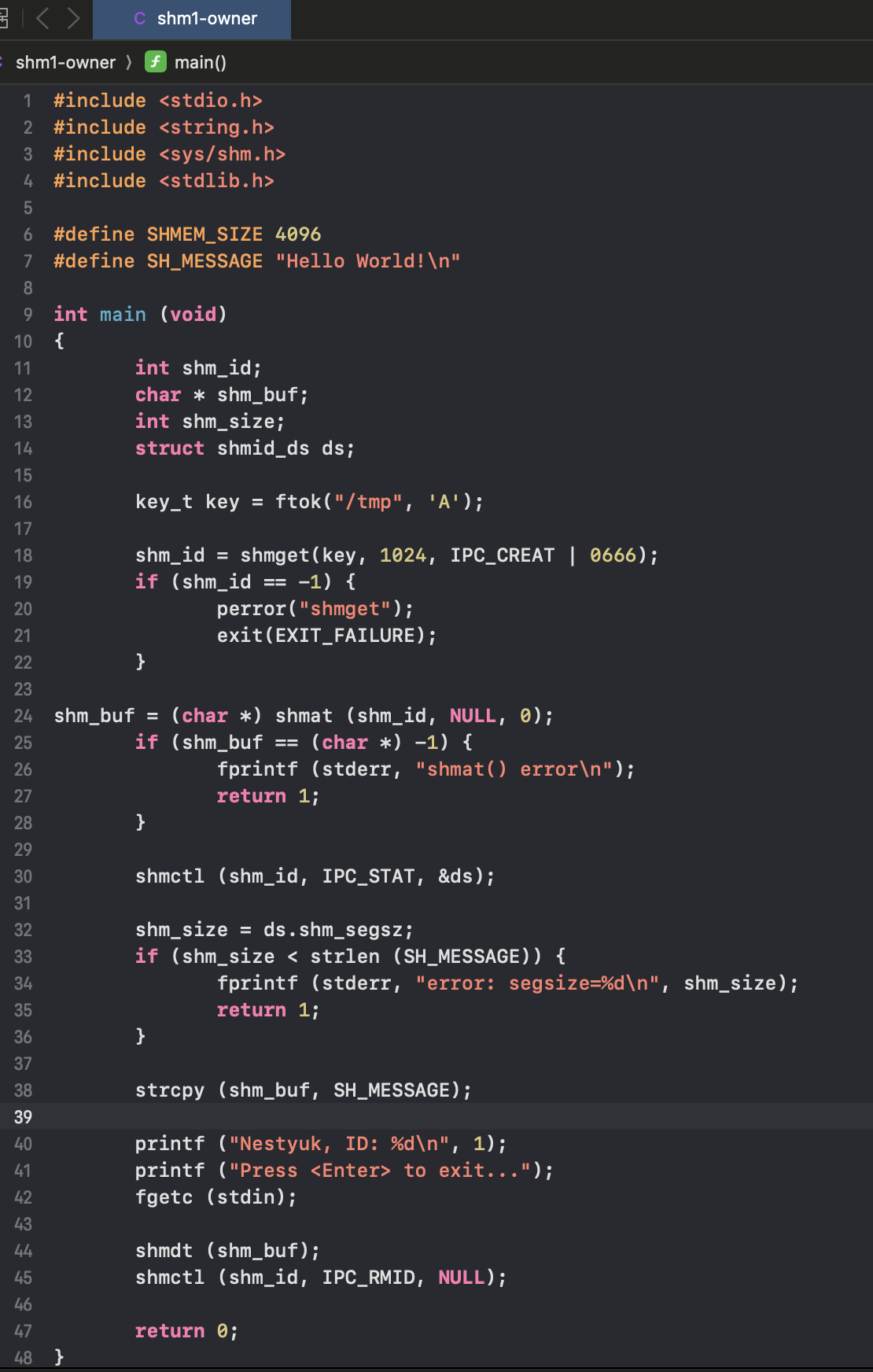
**Цель работы:** Освоить механизм взаимодействия между процессами на основе использования общей памяти.

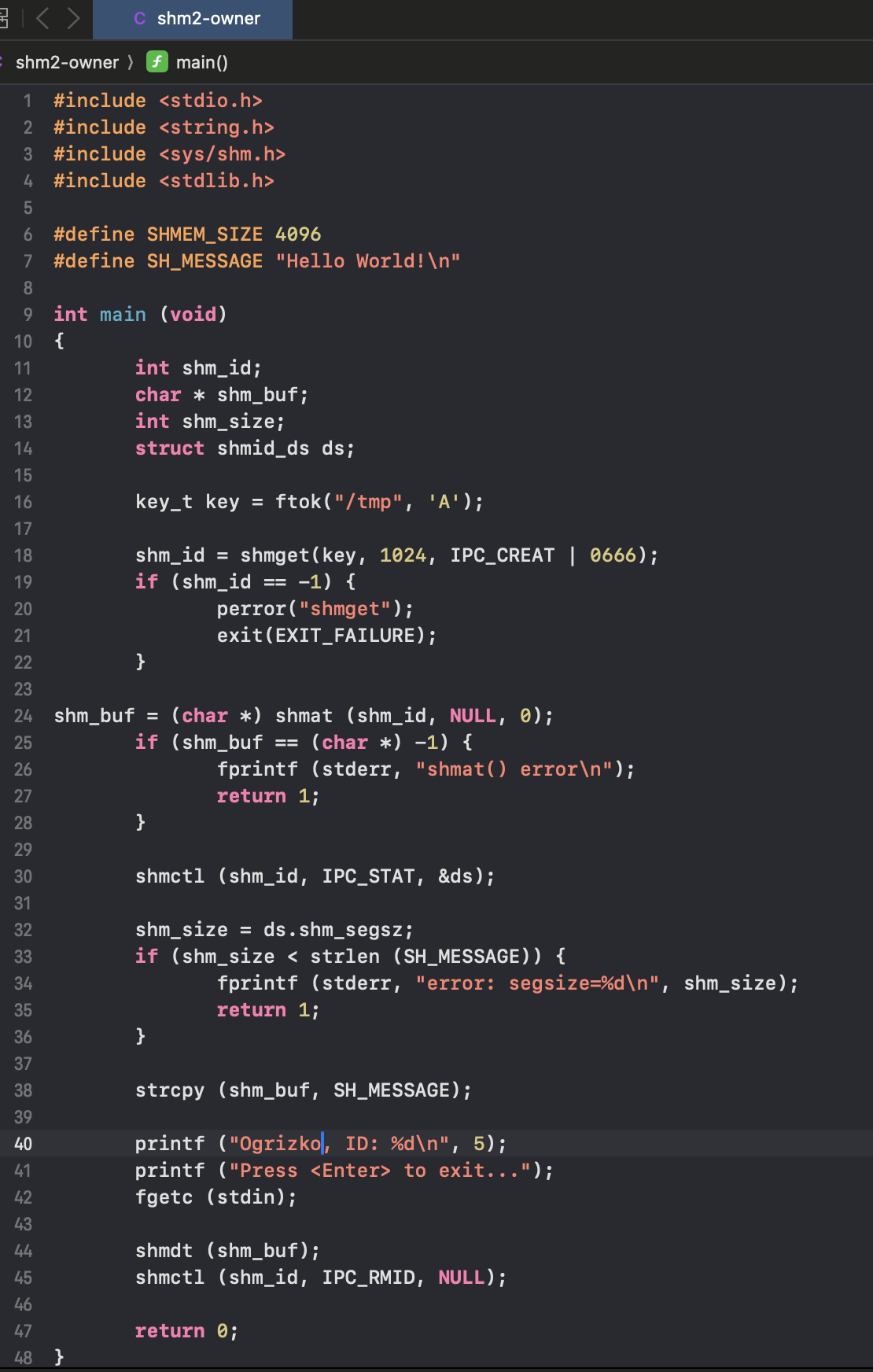
## Задание 1

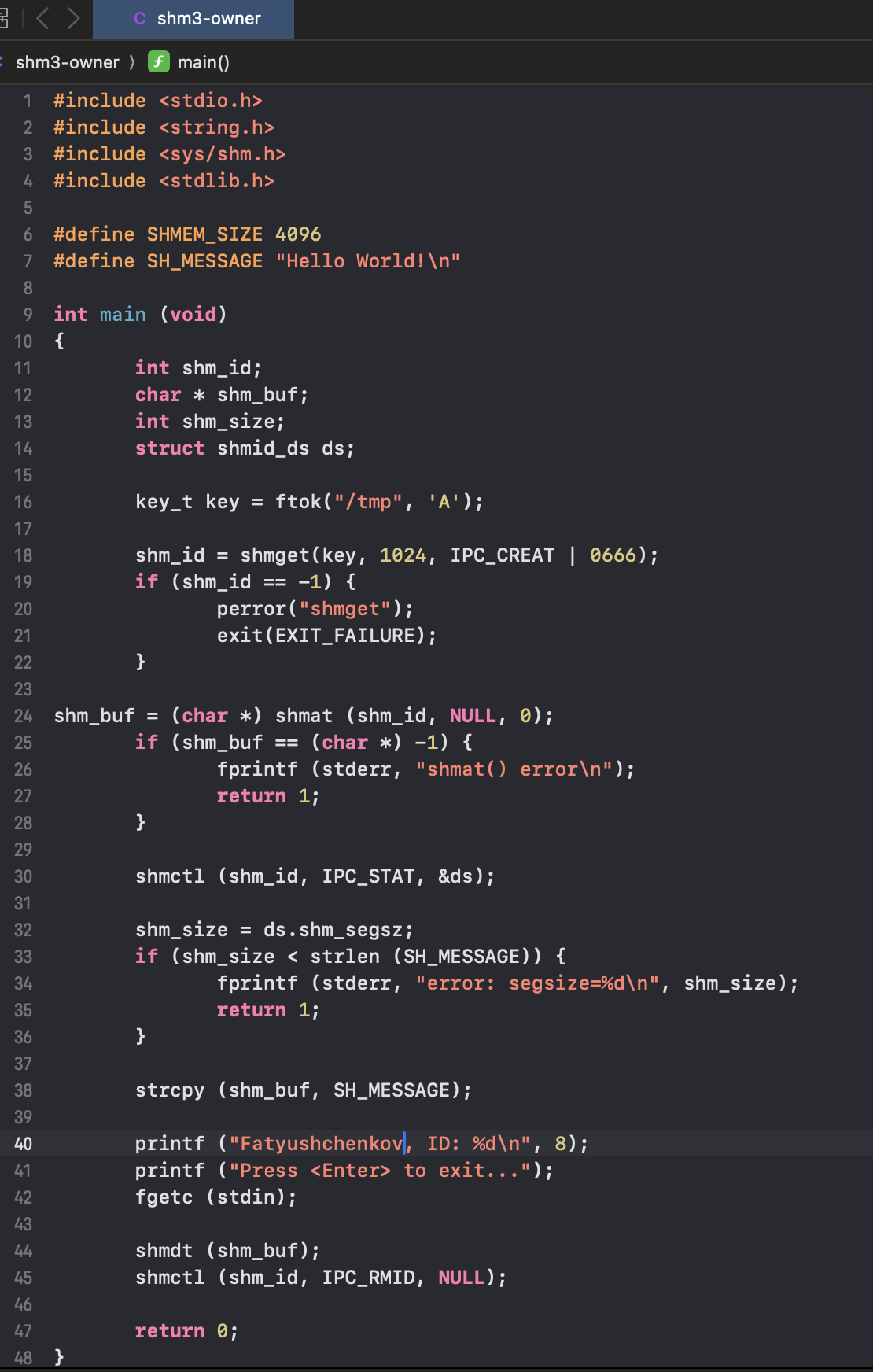
Используя листинги упражнения №1 создайте еще несколько программ **shm1-ownerХХ.c** (количество программ «ownerХХ» должно соответствовать количеству членов команды). Программы должны выводить в сообщениях имена членов команды. Перестройте коды программ так, чтобы процессы обращались к одному сегменту памяти.

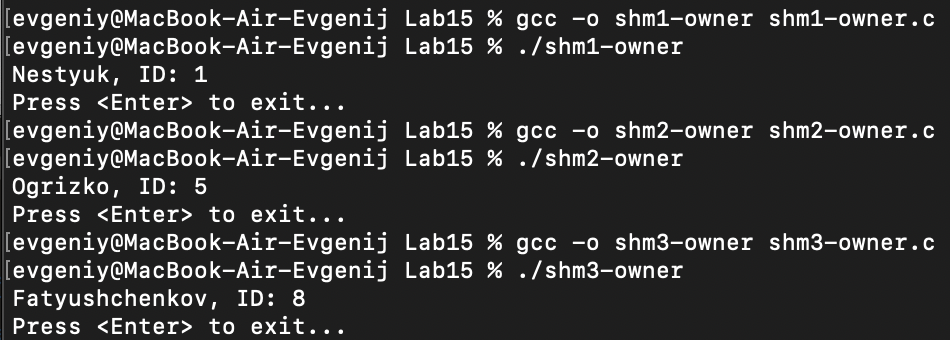
Решение

**Создаем shm-owner.c**









**При помощи метода shmget получаем доступ к сегменту памяти**

## Задание 2.

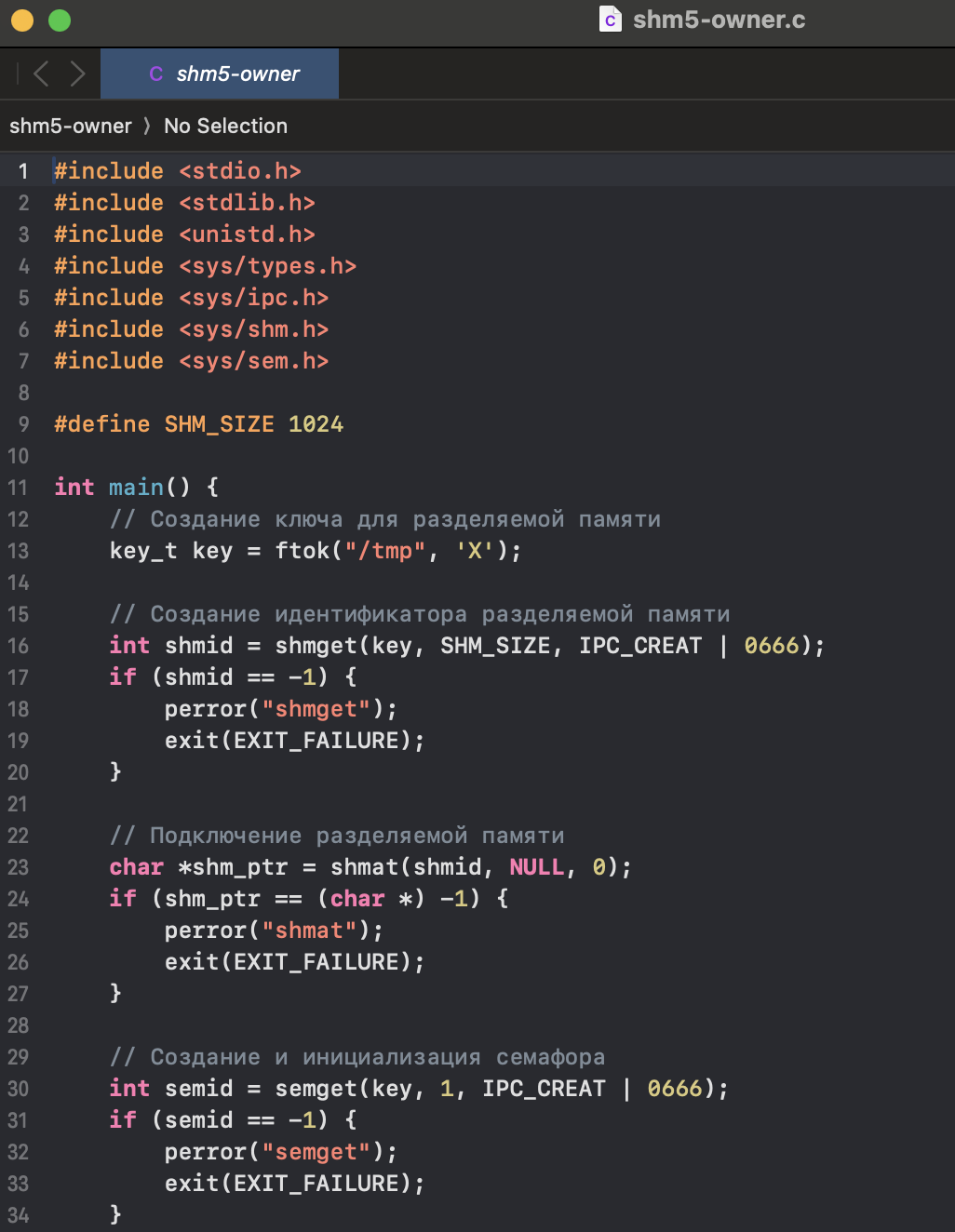
Использования условия задания №1, создайте несколько программ **shm1-ownerХХ.c**, с использованием одного семафора.

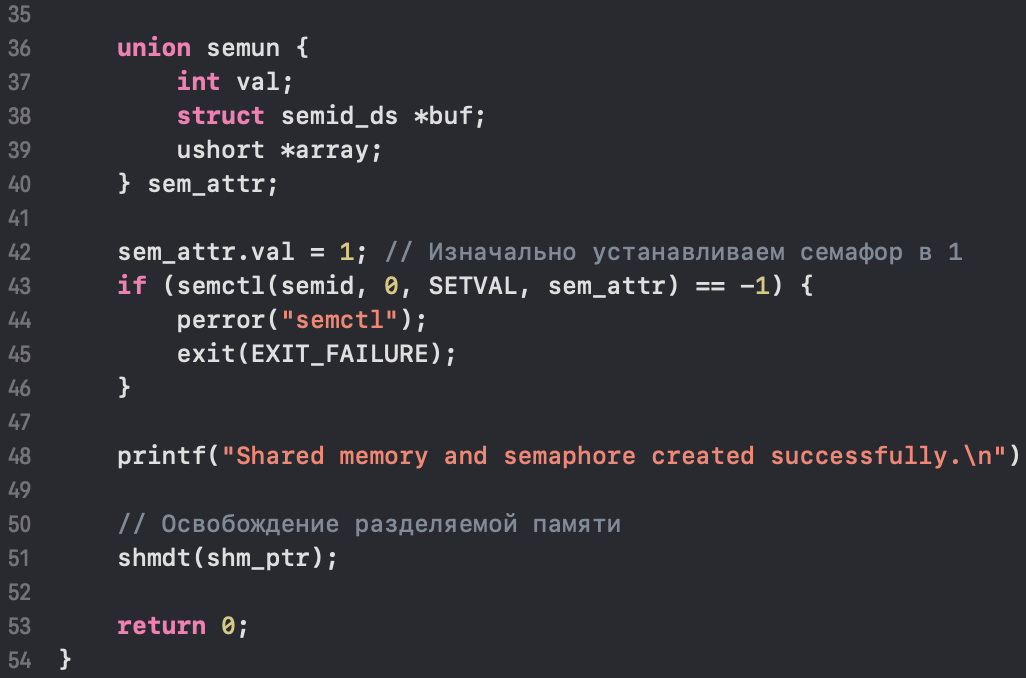
1. Подключить к *общей памяти* двух клиентов (один из них выполняется как поток).

2. Создать два *объекта общие памяти*. Первый процесс пишет в первый *объект* и читает из *второго объекта памяти*, второй процесс, наоборот, читает из первого объекта и пишет во второй (программа-чат).

3. Сделать два сервера сообщений и одного клиента. Серверы используют одну и ту же общую память. Клиент читает и определяет, от какого сервера пришло сообщение.

**Создаем shm5-owner.c**

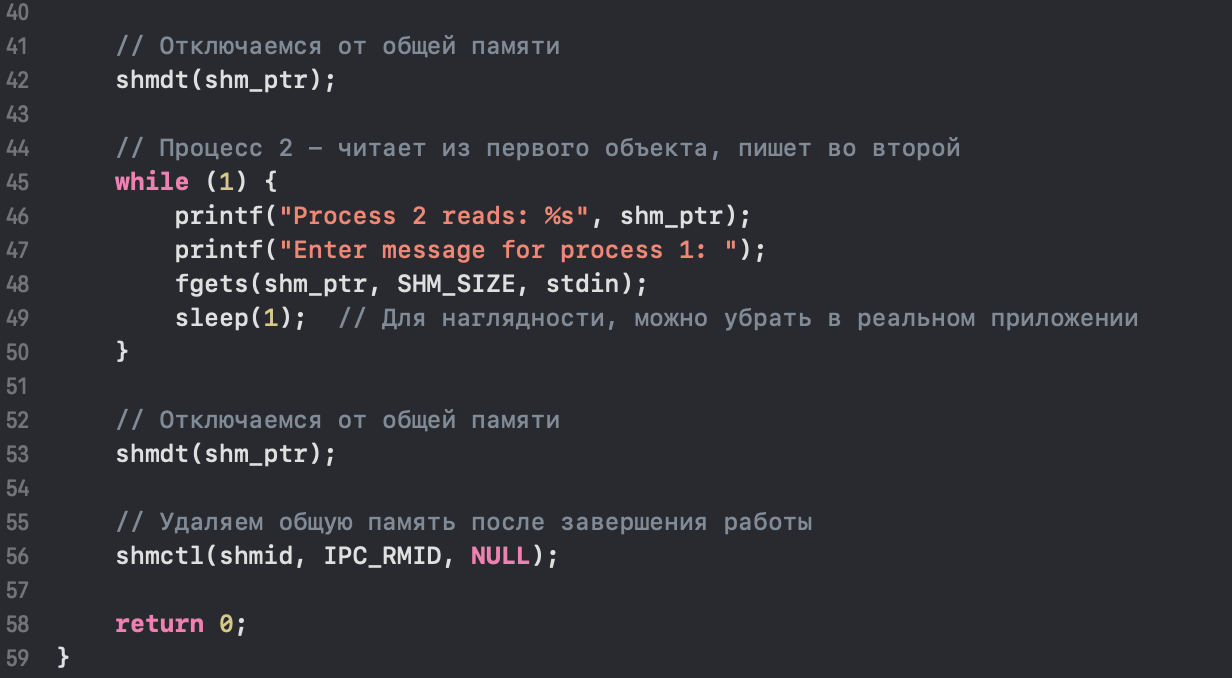


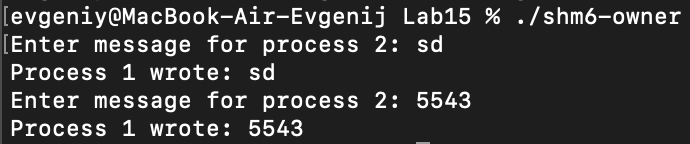




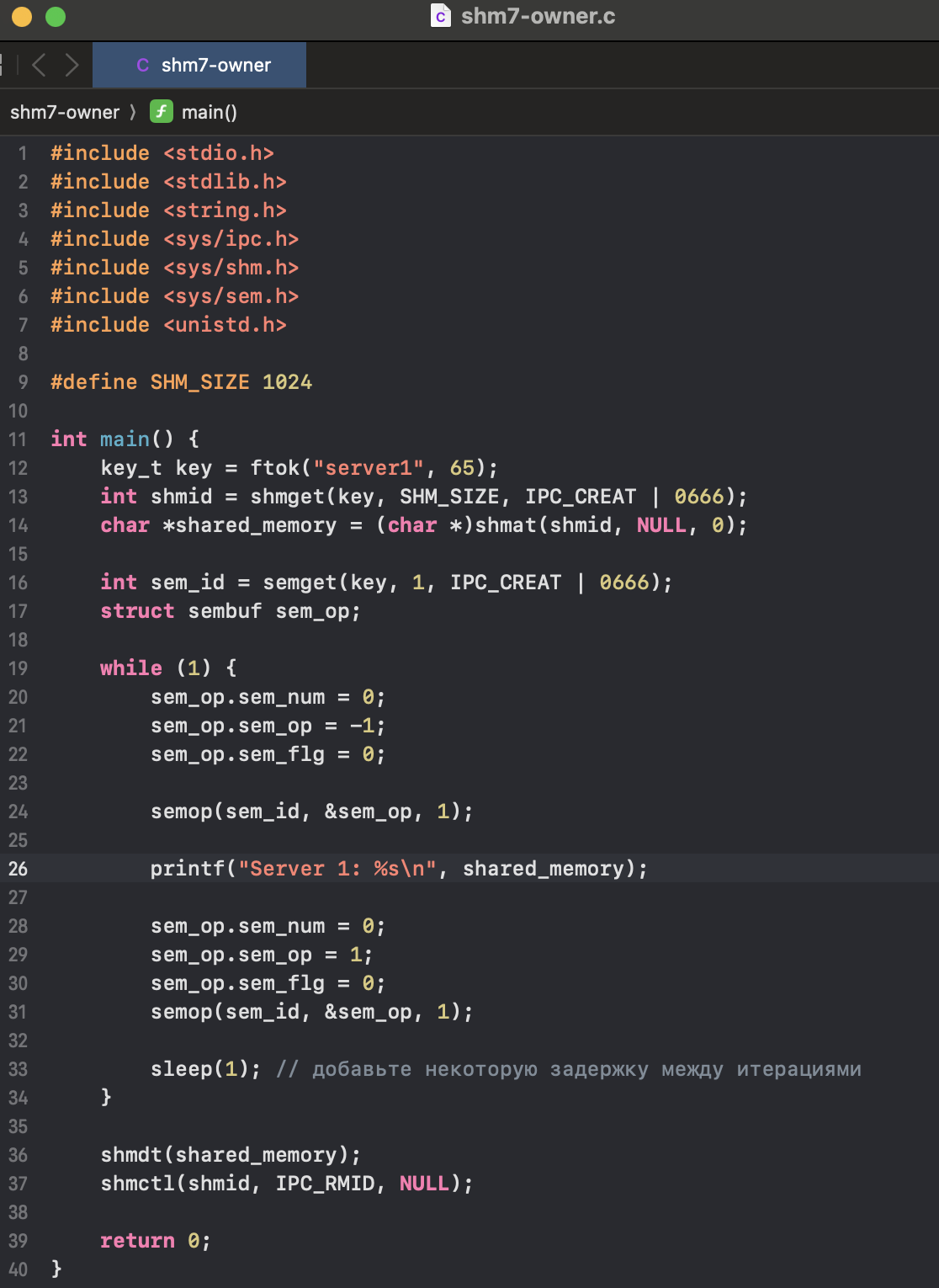
**Создаем shm6-owner.c**

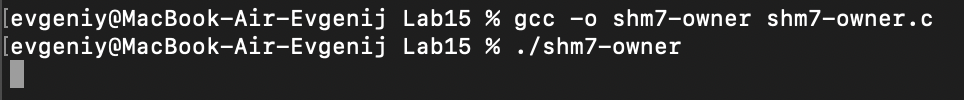






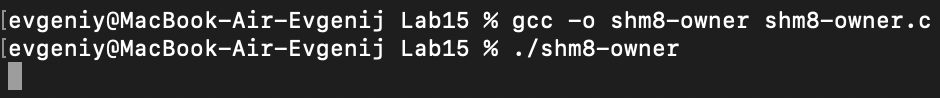
**Создаем shm7-owner.c**

****

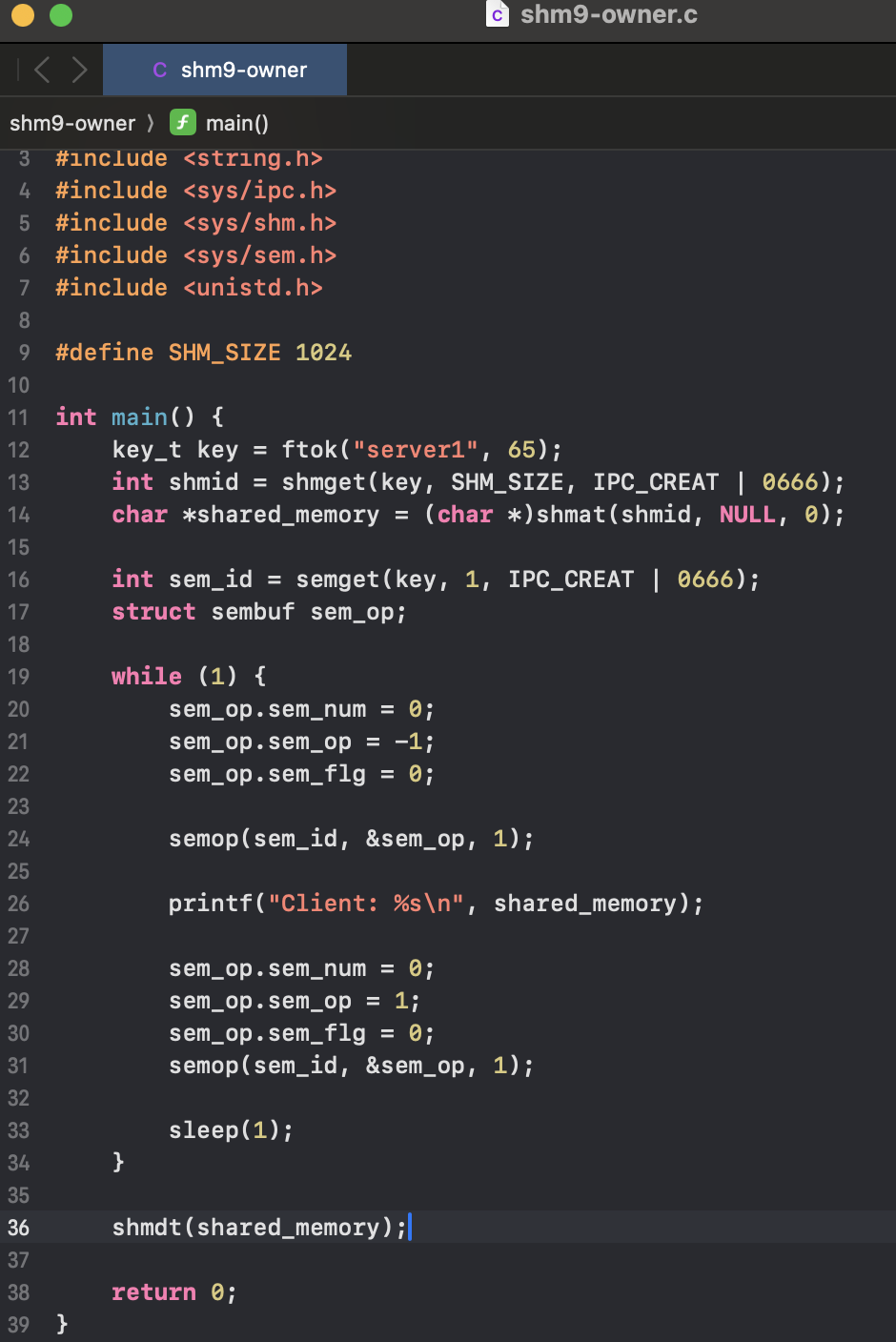
****

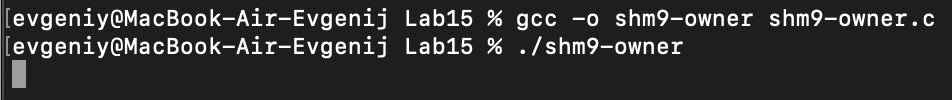
**Создаем shm8-owner.c**





**Создаем shm9-owner.c**





**Вывод:** Освоили механизм взаимодействия между процессами на основе использования общей памяти.

**Контрольные вопросы:**

1. **Что такое *разделяемая память***?

Разделяемая память - это память, которую могут одновременно использовать несколько программ с целью обеспечения связи между ними или избежания избыточных копий.

1. **Что такое *сегмент* памяти, и его атрибуты?**

Сегмент памяти - это логическая единица памяти, которая может быть длиной до 64 килобайт. Каждый сегмент состоит из непрерывных участков памяти. Это независимая, отдельно адресуемая единица.

1. **Что такое *синхронизация процессов*?**

Синхронизация процессов - это координация выполнения нескольких процессов в многопроцессорной системе, чтобы обеспечить контролируемый и предсказуемый доступ к общим ресурсам.

1. **Какие системные вызовы используются для работы с разделяемой памятью? Какие атрибуты они имеют, какие значения возвращают? Какие библиотеки необходимы для их подключения?**

Системные вызовы, используемые для работы с разделяемой памятью, включают shmget(), shmat(), shmdt() и shmctl(). Они используются для создания идентификатора разделяемого сегмента памяти, присоединения этого сегмента к адресному пространству процесса, отсоединения сегмента от адресного пространства и управления разделяемыми сегментами памяти.

1. **Что такое *семафоры*, для чего они используются?**

Семафоры - это обычные переменные, используемые для координации действий нескольких процессов в компьютерной системе. Они используются для обеспечения взаимного исключения, избегания состояний гонки и реализации синхронизации между процессами.

1. **Какие системные вызовы используются для работы с семафорами? Какие атрибуты они имеют, какие значения возвращают?**

Системные вызовы, используемые для работы с семафорами, включают операции ожидания (P) и сигнала (V). Операция ожидания уменьшает значение семафора, а операция сигнала увеличивает его. Когда значение семафора равно нулю, любой процесс, выполняющий операцию ожидания, будет заблокирован, пока другой процесс не выполнит операцию сигнала.

1. **Опишите примерный сценарий работы с разделяемой памятью с использованием семафоров.**

Работа с разделяемой памятью с использованием семафоров обычно включает в себя следующие шаги: процесс, который хочет войти в критическую секцию, вызывает операцию ожидания на семафоре. Если значение семафора больше нуля, процесс продолжает свое выполнение; в противном случае процесс блокируется. После завершения работы в критической секции процесс вызывает операцию сигнала на семафоре, освобождая его для других процессов.

1. **Что Вы знаете о синхронизации и семафорах в языках более высокого уровня *C#* или *Java*?**

В языках высокого уровня, таких как C# и Java, семафоры и синхронизация также используются для координации действий многопоточных программ. Они обеспечивают контролируемый доступ к общим ресурсам и предотвращают состояния гонки.