**Филиал федерального государственного бюджетного образовательного учреждения высшего образования**

**«Национальный исследовательский университет «МЭИ»**

**в г. Смоленске**

Кафедра вычислительной техники

Направление: 09.04.01. «Информатика и вычислительная техника»

Профиль: «Программное обеспечение средств вычислительной техники и

автоматизированных систем»

Практическая работа №3

«Отладка на кластере простых программ с использованием MPI»

по курсу:

«Вычислительные системы»

Студент: Старостенков А.А.

Группа: ВМ-22(маг)

Вариант: 19

Преподаватель: Федулов А.С.

Смоленск, 2023

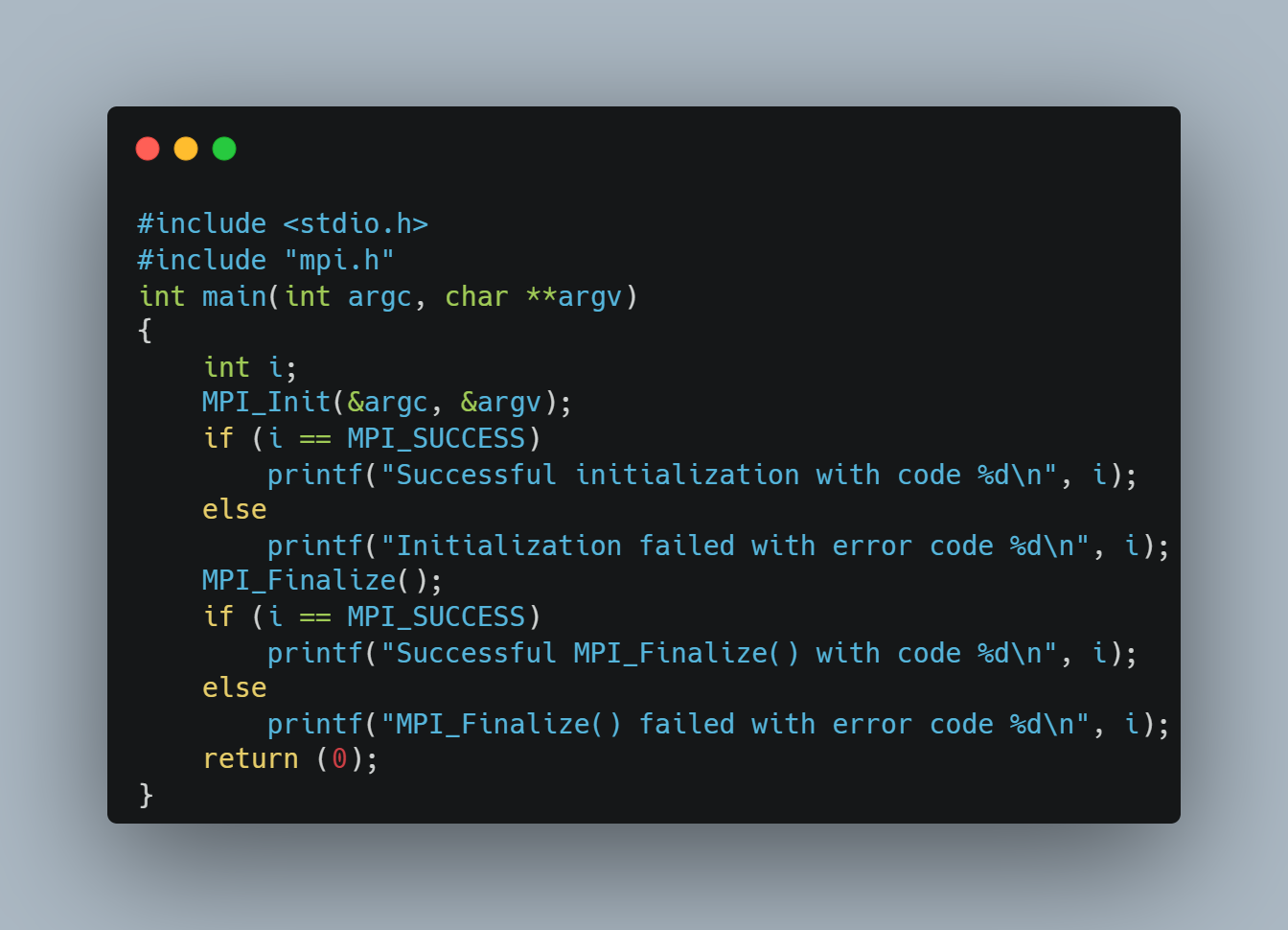
**Задание**

1. Отладить, скомпилировать и запустить на гибридном вычислительном кластере (ГВК) СФМЭИ простые тестовые программы на языке Си с использованием технологии MPI. В качестве тестовых программ использовать программы 1- 8 из лекции 4 по курсу «Вычислительные системы».
2. Программу 1 запустить на текущем узле с числом процессов, равным **номеру по журналу+2**.
3. Программу 2 запустить на текущем узле с числом процессов, равным **номеру по журналу+1**.
4. Программу 3 запустить с числом процессов, равным **номеру по журналу+3** на одном, двух и трех узлах. Запустить программу 3 на одном, двух и трех узлах с максимальным числом процессов.
5. Программу 4 запустить несколько раз с числом процессов, равным **номеру по журналу+4.** Убедиться, что результат меняется от запуска к запуску.
6. Программу 5 запустить на текущем узле с числом процессов, равным **номеру по журналу+5**.
7. Программу 6 запустить с числом процессов, равным **10.** Модифицируйте программу 6 таким образом, чтобы в результате выполнения обмена сообщениями процессы 2 и 3 содержали следующие значения: **a=номер по журналу, b=номер по журналу+1**.
8. Программу 7 запустить с числом процессов, равным **10.** Убедиться в том, что показания таймера могут быть различными в разных процессах.
9. Программу 8 запустить с числом процессов, равным **10.**
10. Результат выполнения каждого примера дополнить кратким описанием (анализом, объяснением).
11. Выполнение примеров показать преподавателю.
12. Все выполненные задания оформить в виде отчета.
13. Отчет должен содержать: тексты примеров, команды компиляции и запуска, результаты запуска, объяснение результатов.

**Ход работы**

1. Программу 1 запустить на текущем узле с числом процессов, равным **номеру по журналу+2**.

Код



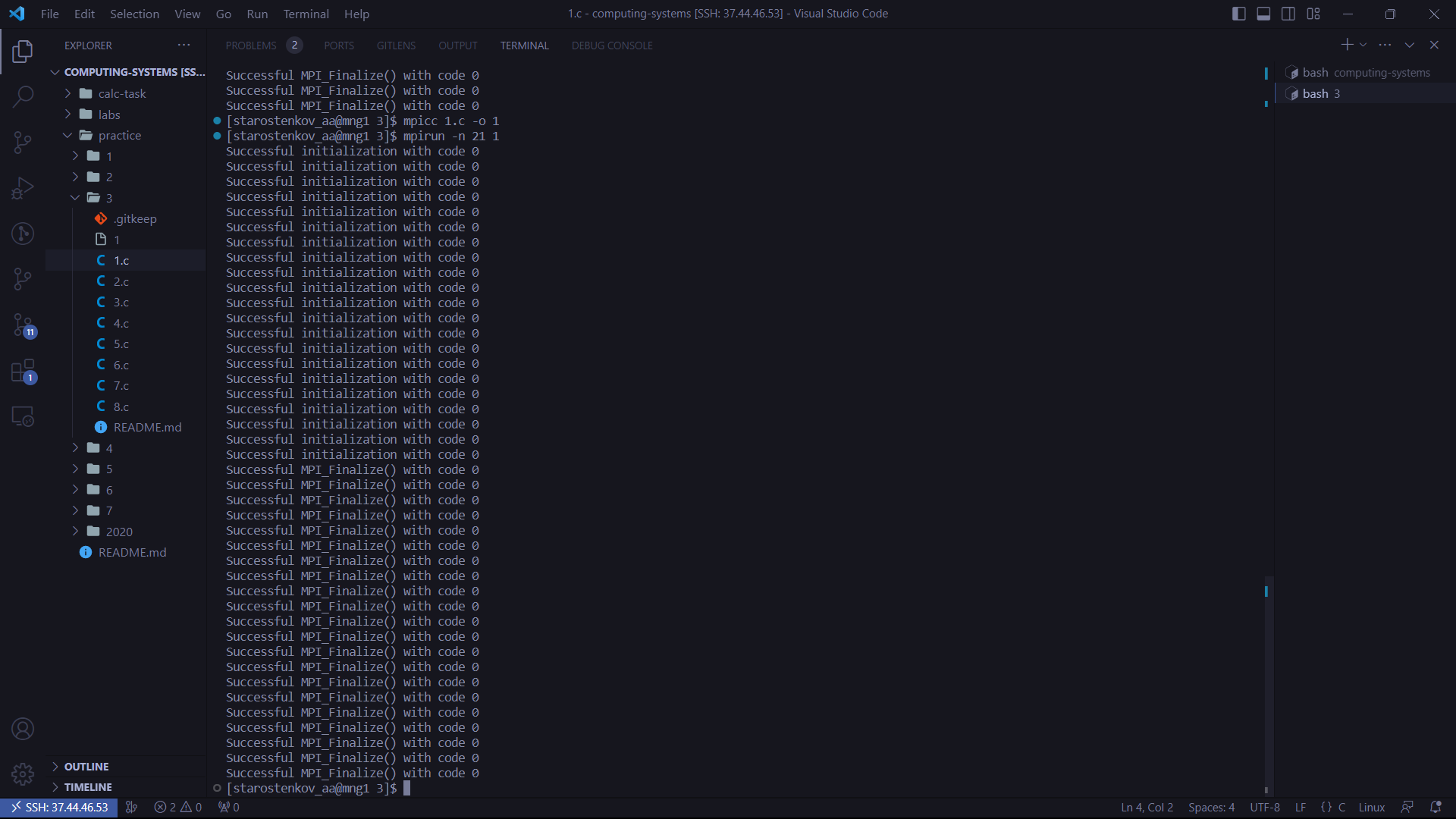


Рисунок 1 – Запуск с 21 процессором

Код выполнился штатно, успешно выведено по 21 строки инициализации.

1. Программу 2 запустить на текущем узле с числом процессов, равным **номеру по журналу+1**.

Код



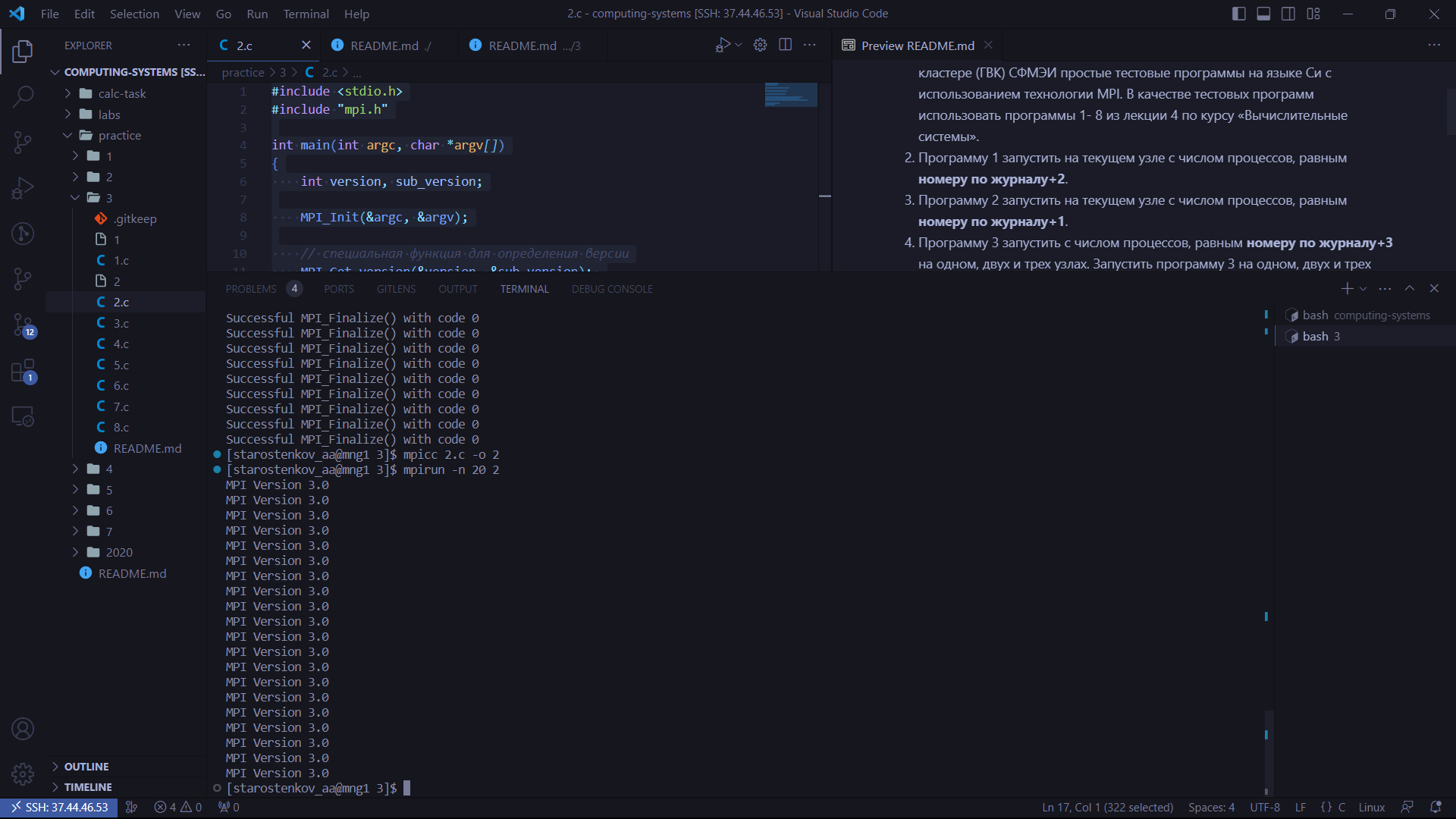


Рисунок 2 – вывод версии

Запускается 20 процессов, которые выводят версию MPI.

1. Программу 3 запустить с числом процессов, равным **номеру по журналу+3** на одном, двух и трех узлах. Запустить программу 3 на одном, двух и трех узлах с максимальным числом процессов.

Код



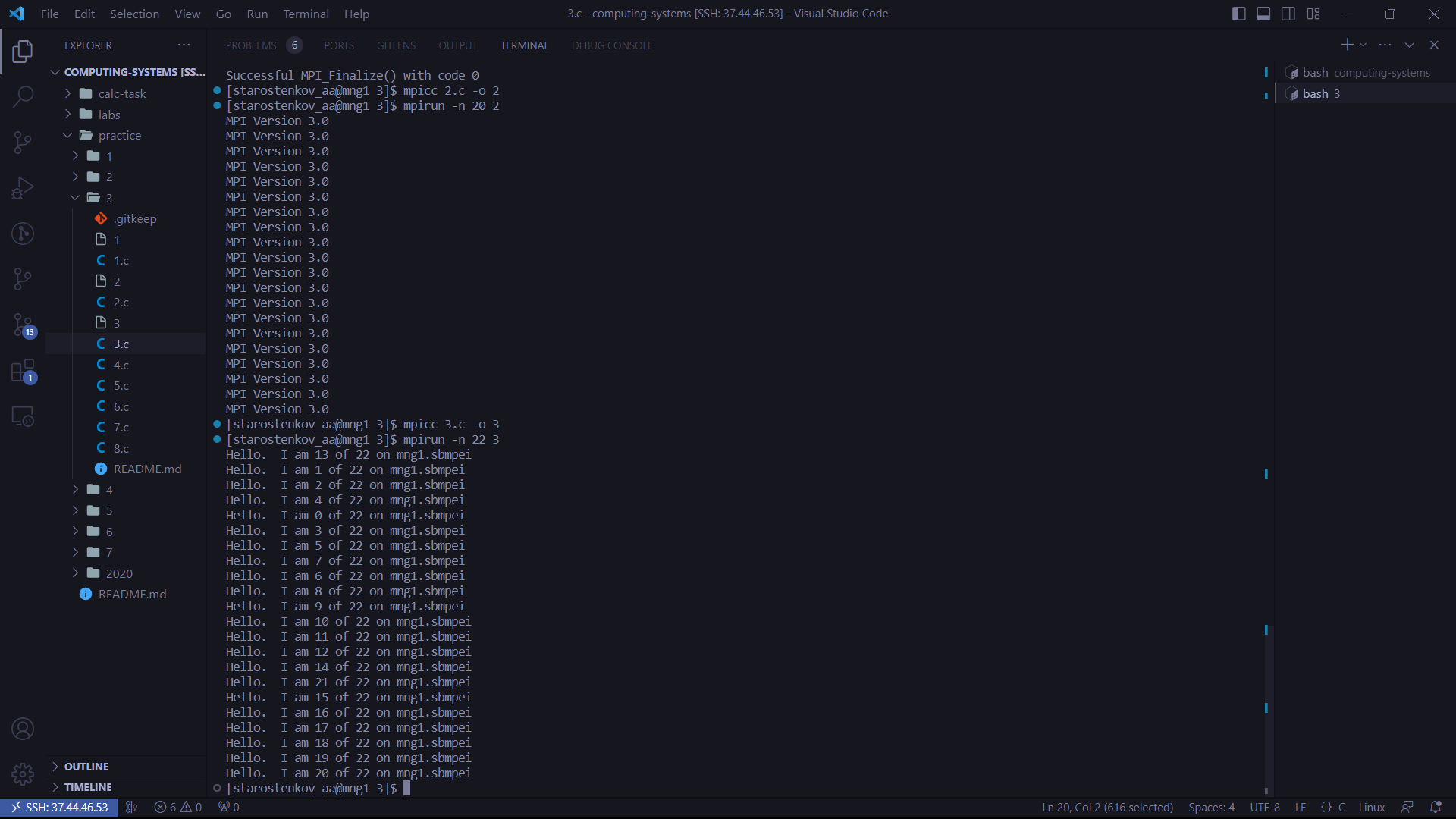


Рисунок 3 – запуска на mng1

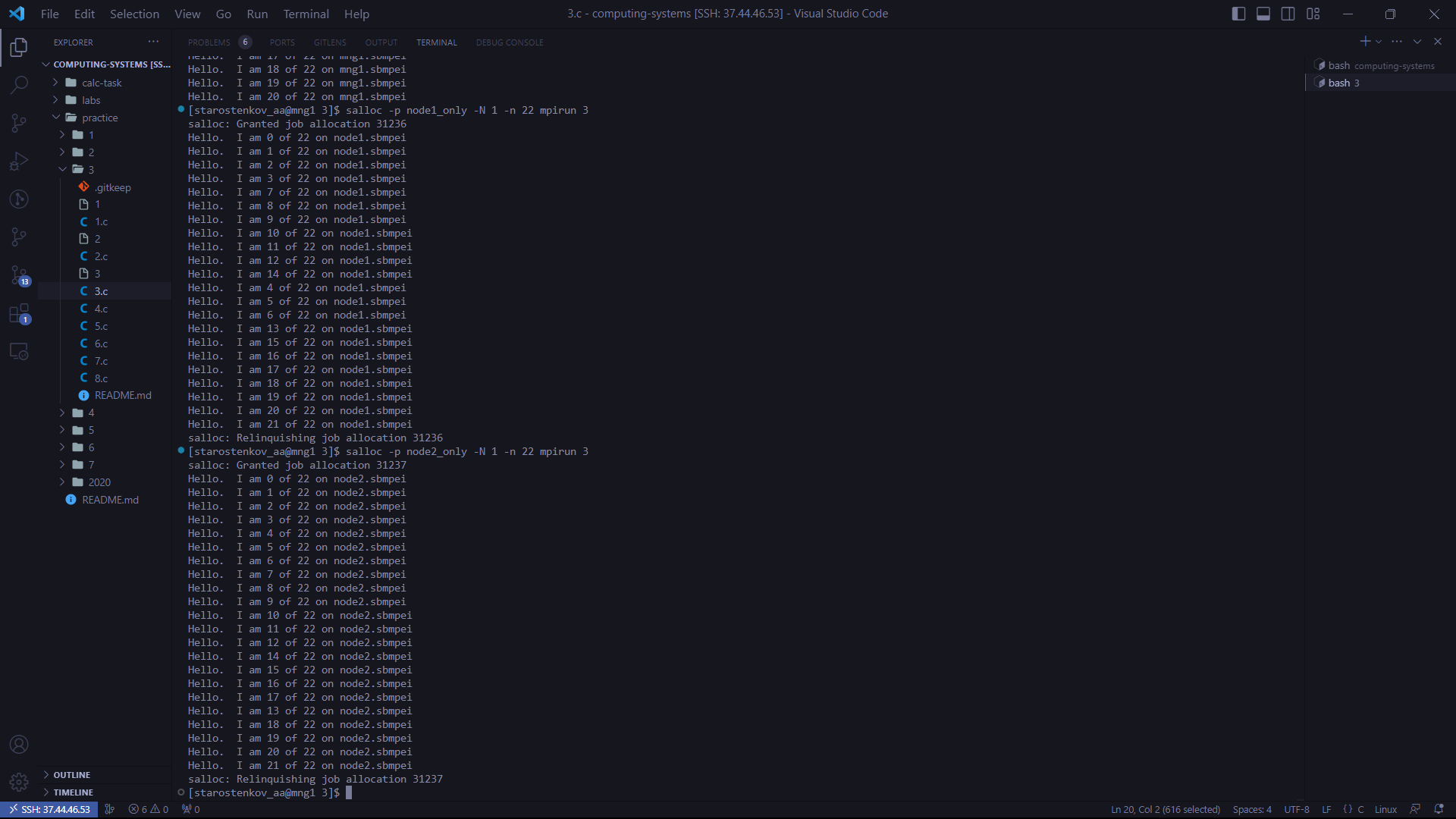


Рисунок 4 – запуск на node1 и node2 по отдельности

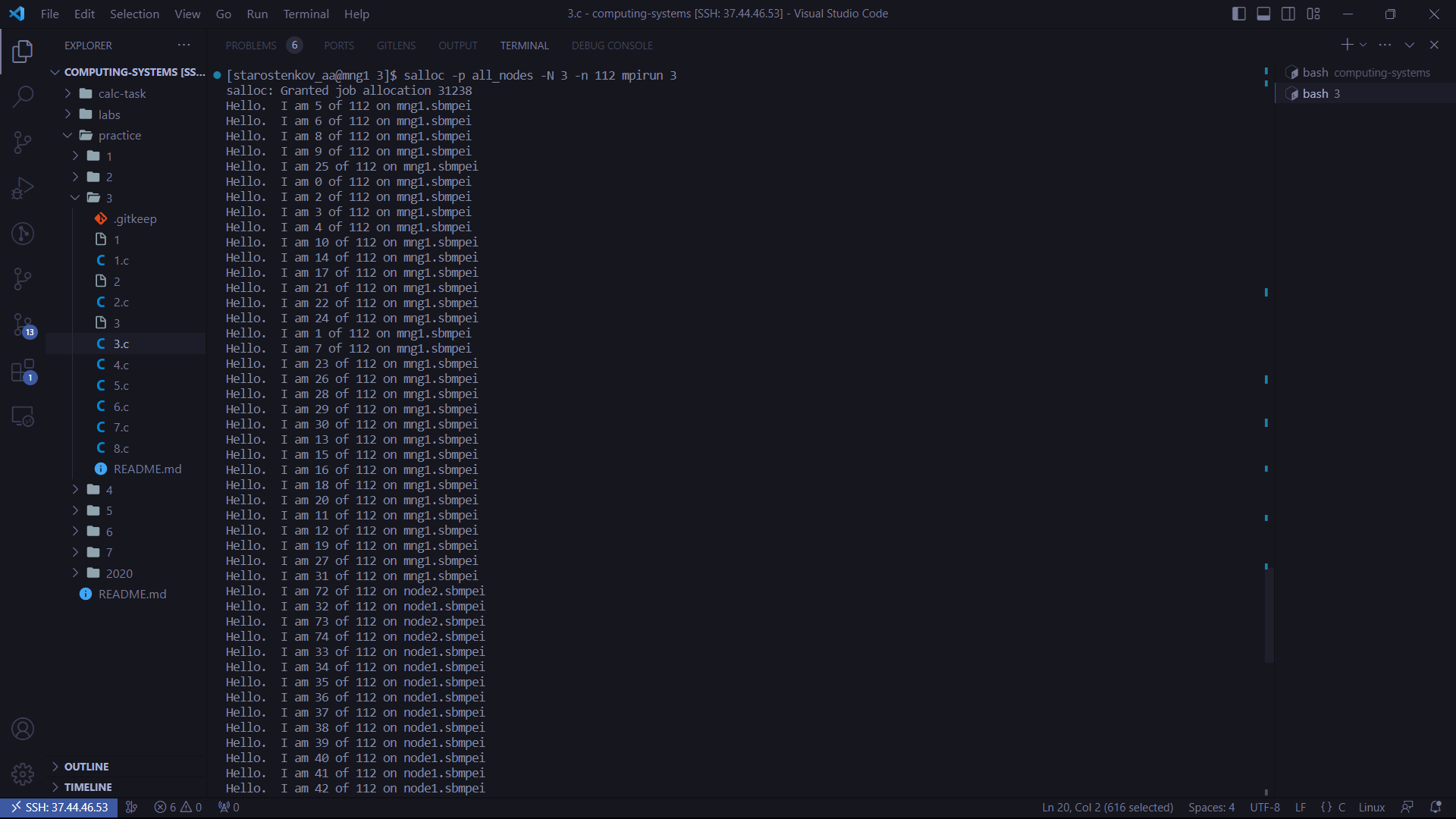
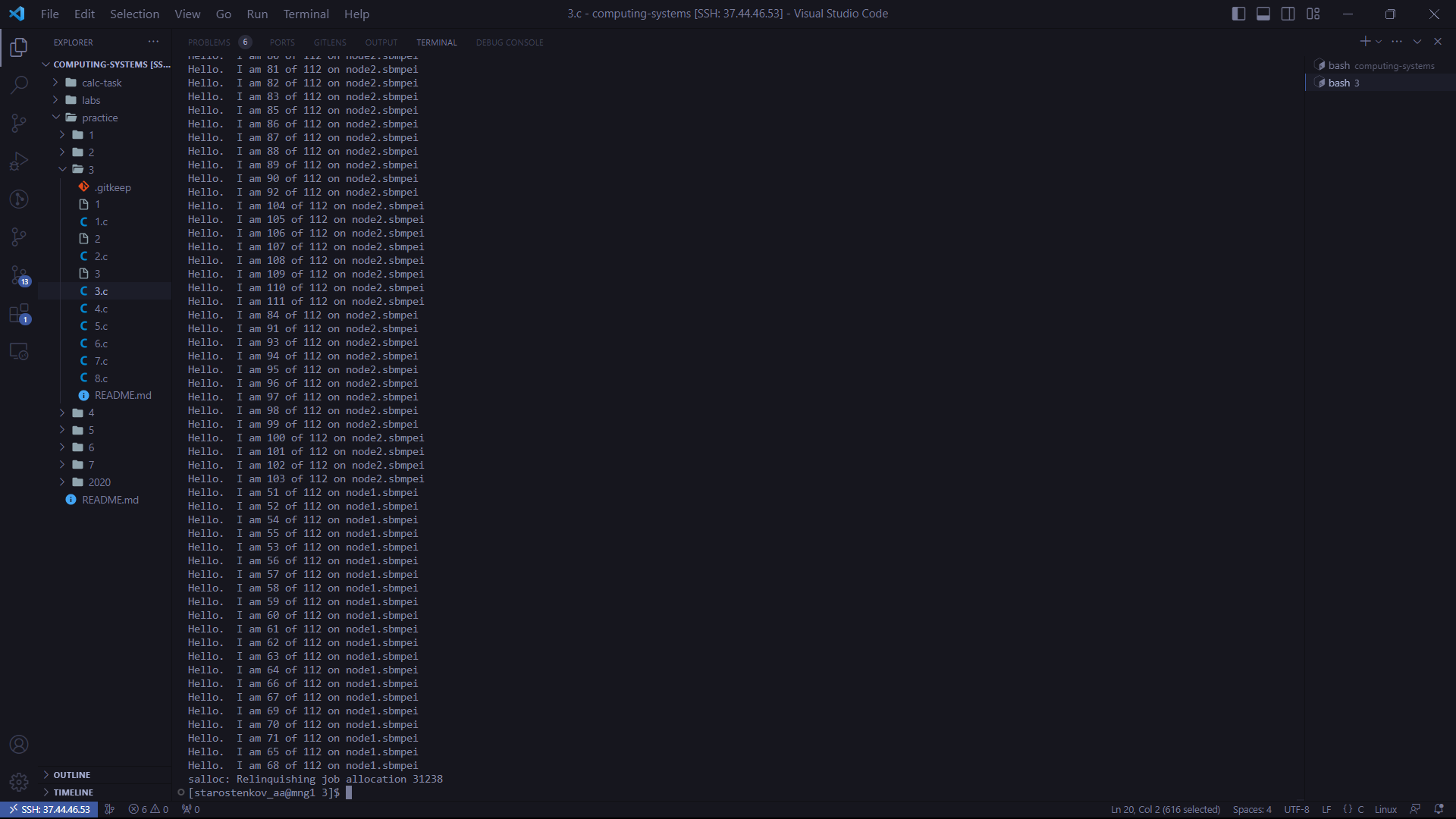
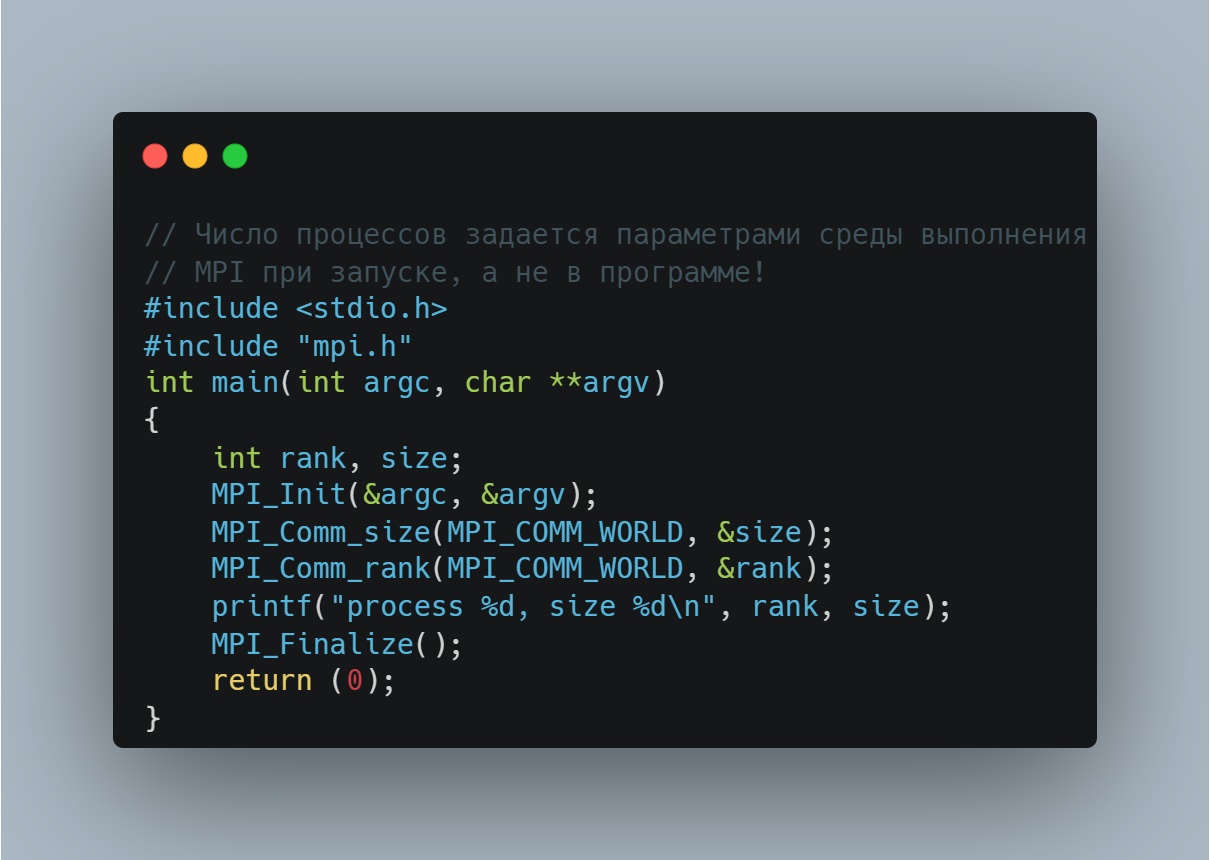
 

Рисунок 5 – запуск all\_nodes

Запуск по узлам выполняется штатно.

1. Программу 4 запустить несколько раз с числом процессов, равным **номеру по журналу+4.** Убедиться, что результат меняется от запуска к запуску.

Код



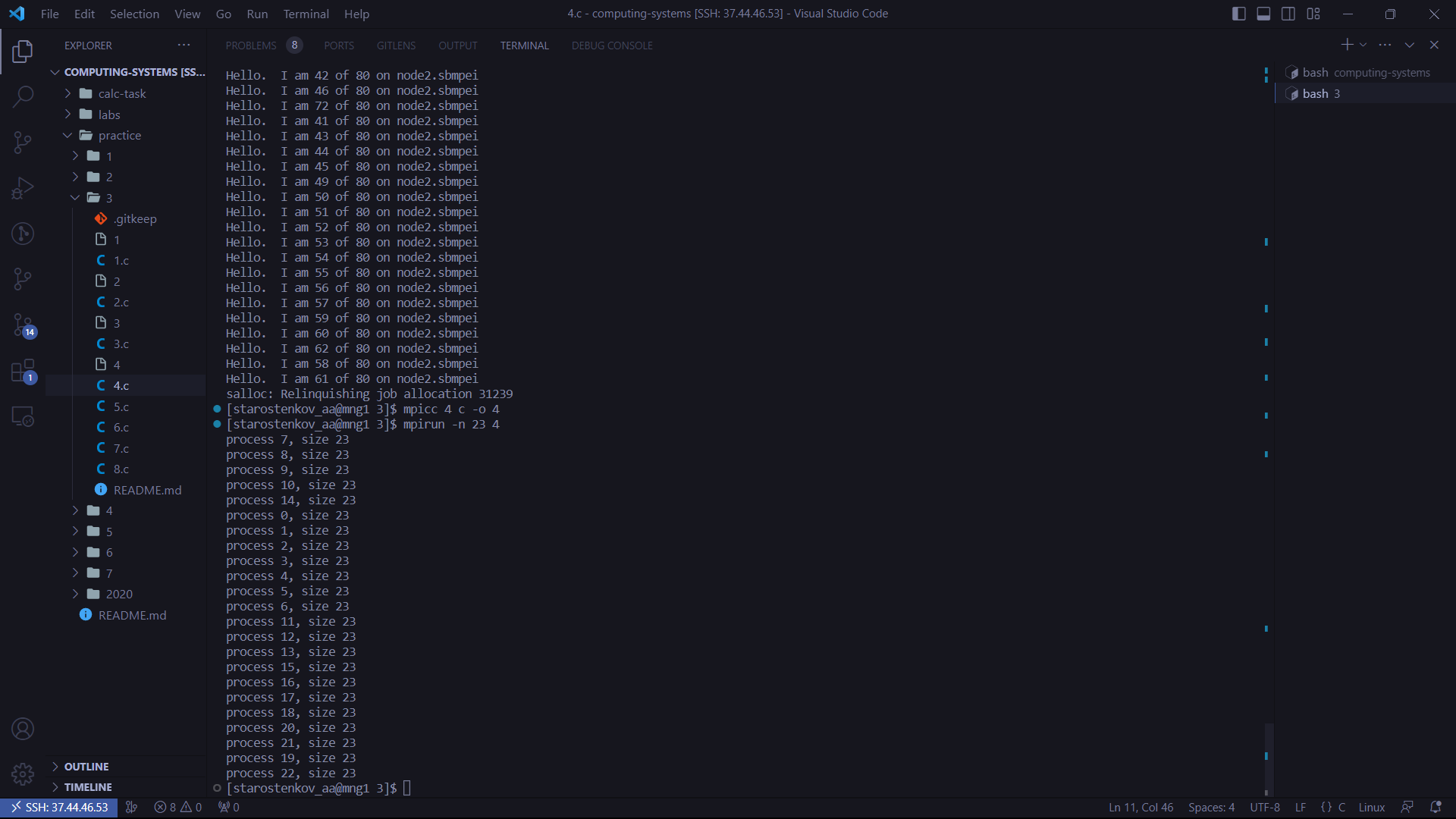
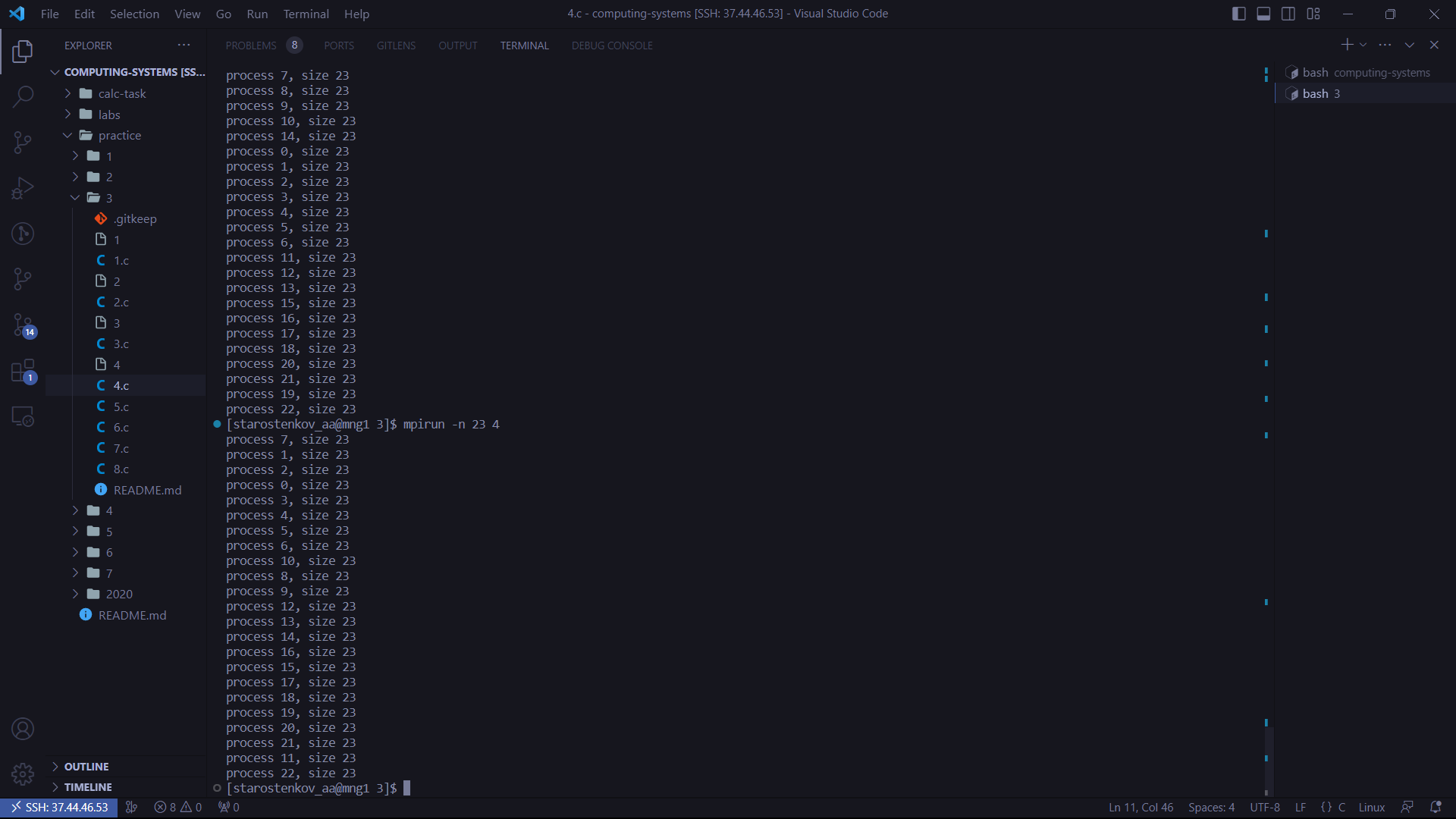
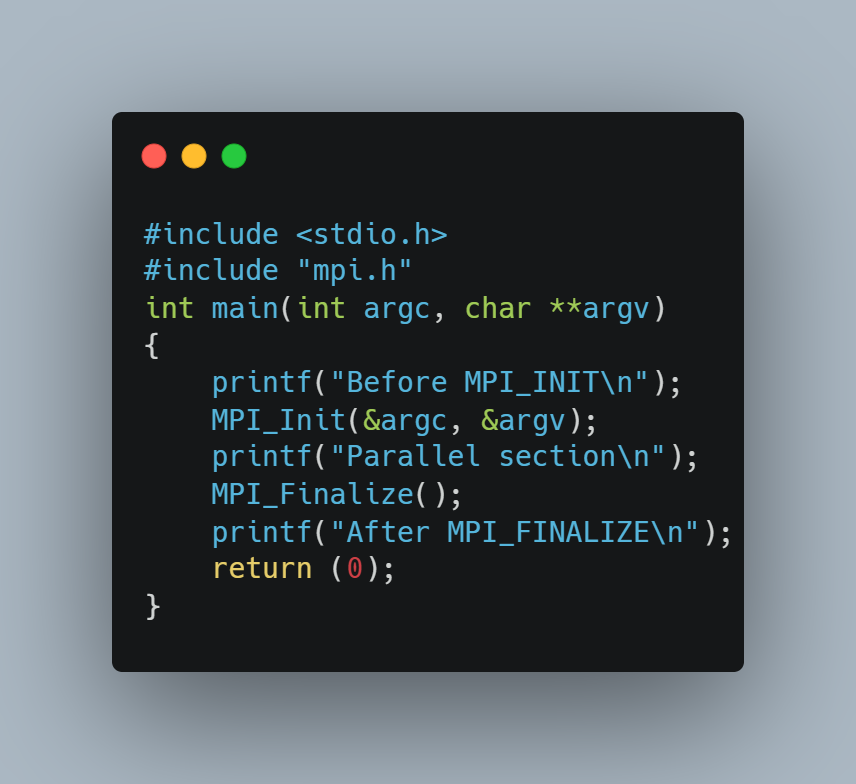
 

Рисунок 6 – процессы

Порядок выполнения процессов является недетерминированным, поэтому вывод программы меняется от запуска к запуску.

1. Программу 5 запустить на текущем узле с числом процессов, равным **номеру по журналу+5**.

Код



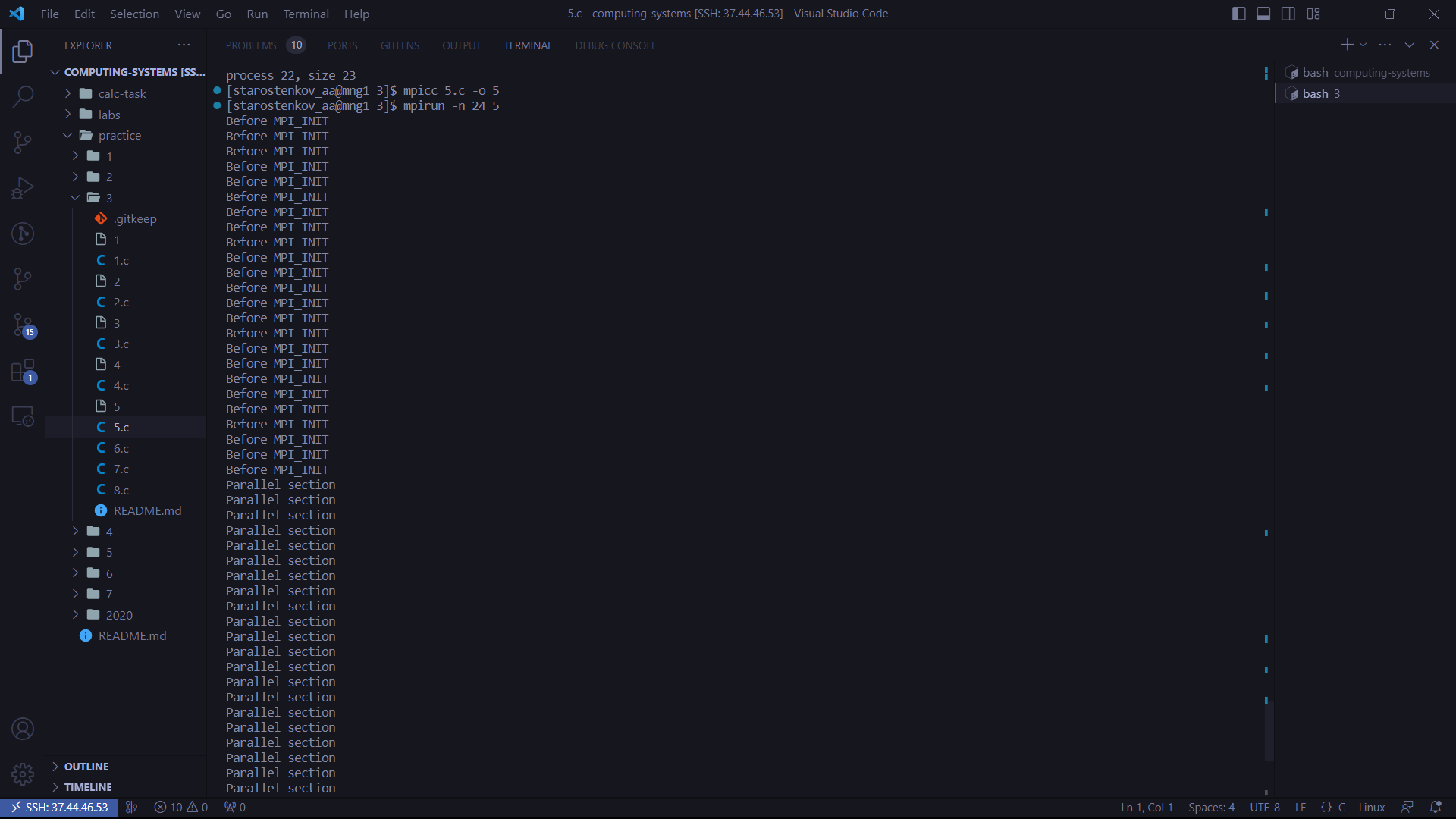
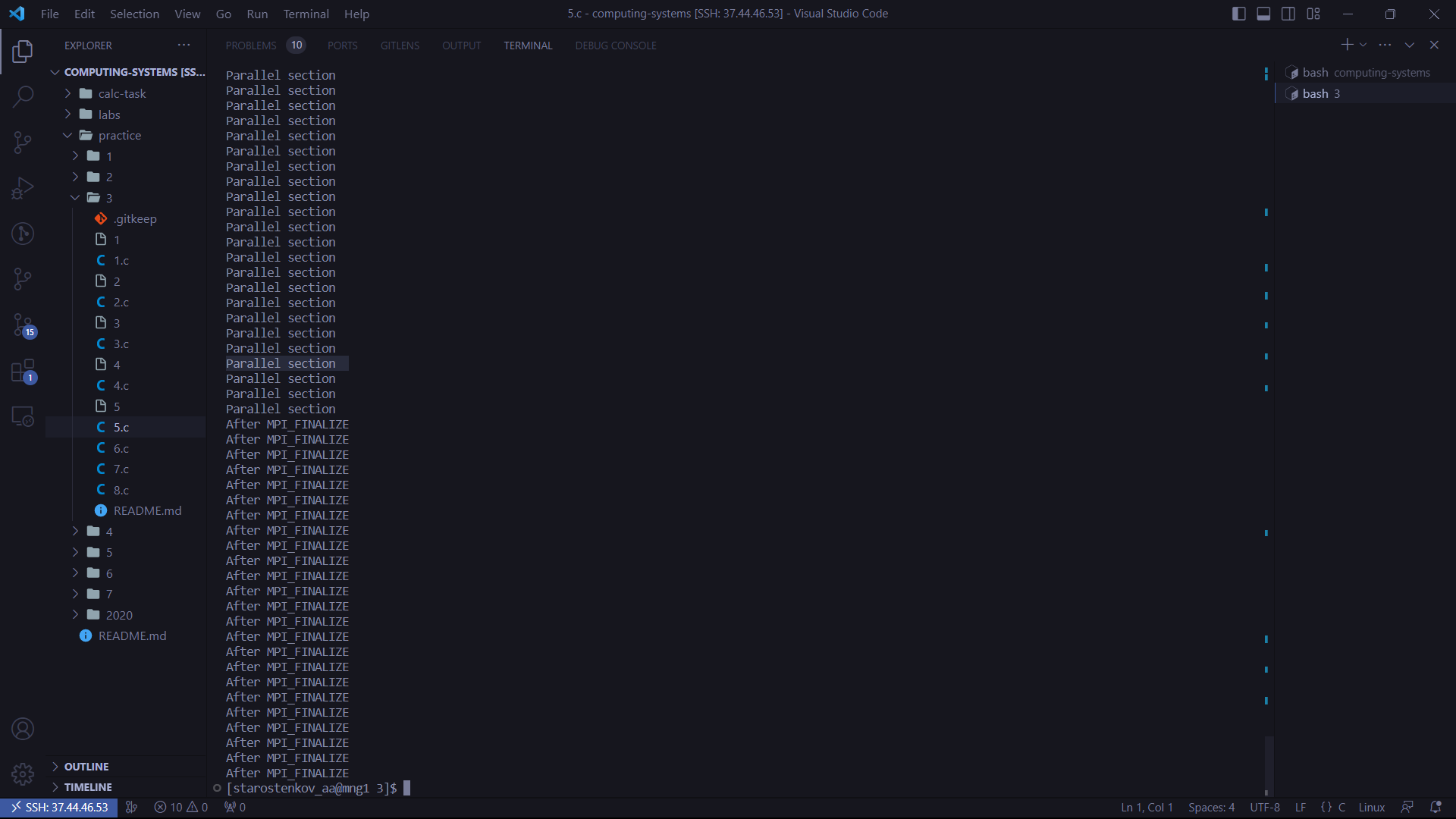
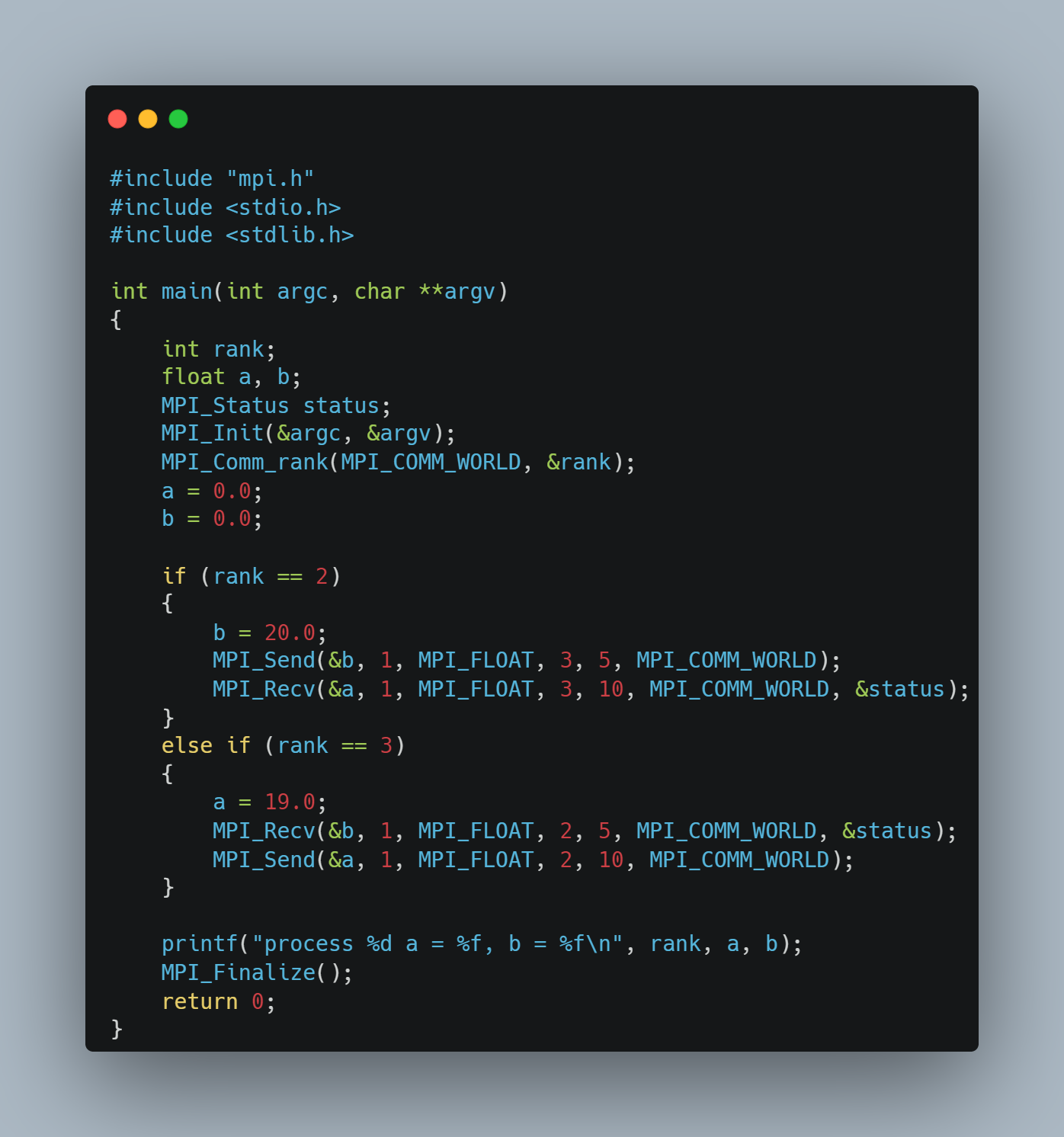
 

Рисунок 7 – Параллельное выполнение

Было выполнено по 3 блока, каждый одновременно имел 24 процесса, что наблюдается на рисунке.

1. Программу 6 запустить с числом процессов, равным **10.** Модифицируйте программу 6 таким образом, чтобы в результате выполнения обмена сообщениями процессы 2 и 3 содержали следующие значения: **a=номер по журналу, b=номер по журналу+1**.

Код



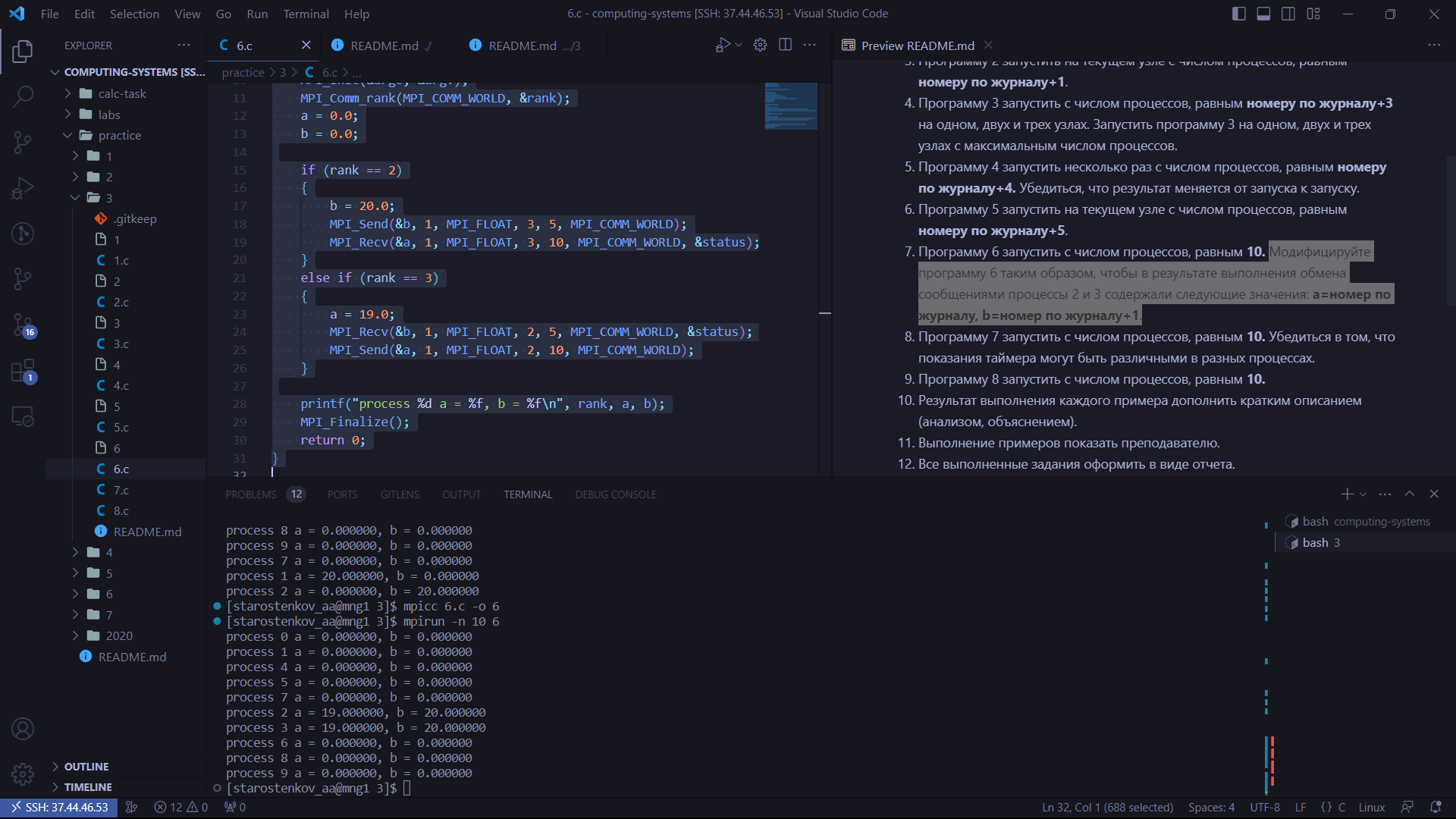
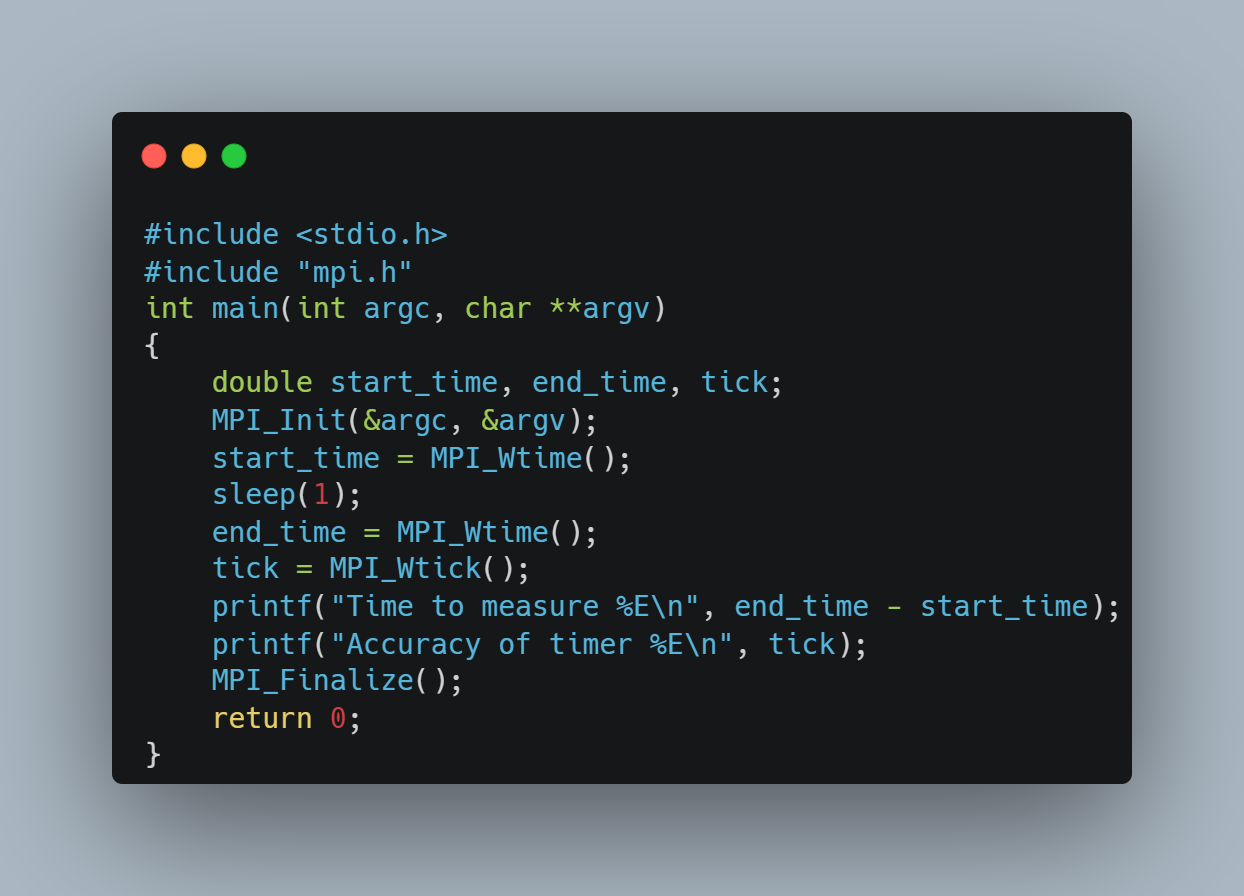


Рисунок 8 – модифицированная программа

Процесс 2 задаёт значение b=20 и отправляет его процессу 3, затем получает значение a от процесса 3. Процесс 3 получает значение b, затем задаёт значение a=19 и отправляет его процессу 2. В результате, процессы 2 и 3 содержат следующие значения: a=19, b=20.

1. Программу 7 запустить с числом процессов, равным **10.** Убедиться в том, что показания таймера могут быть различными в разных процессах.

Код



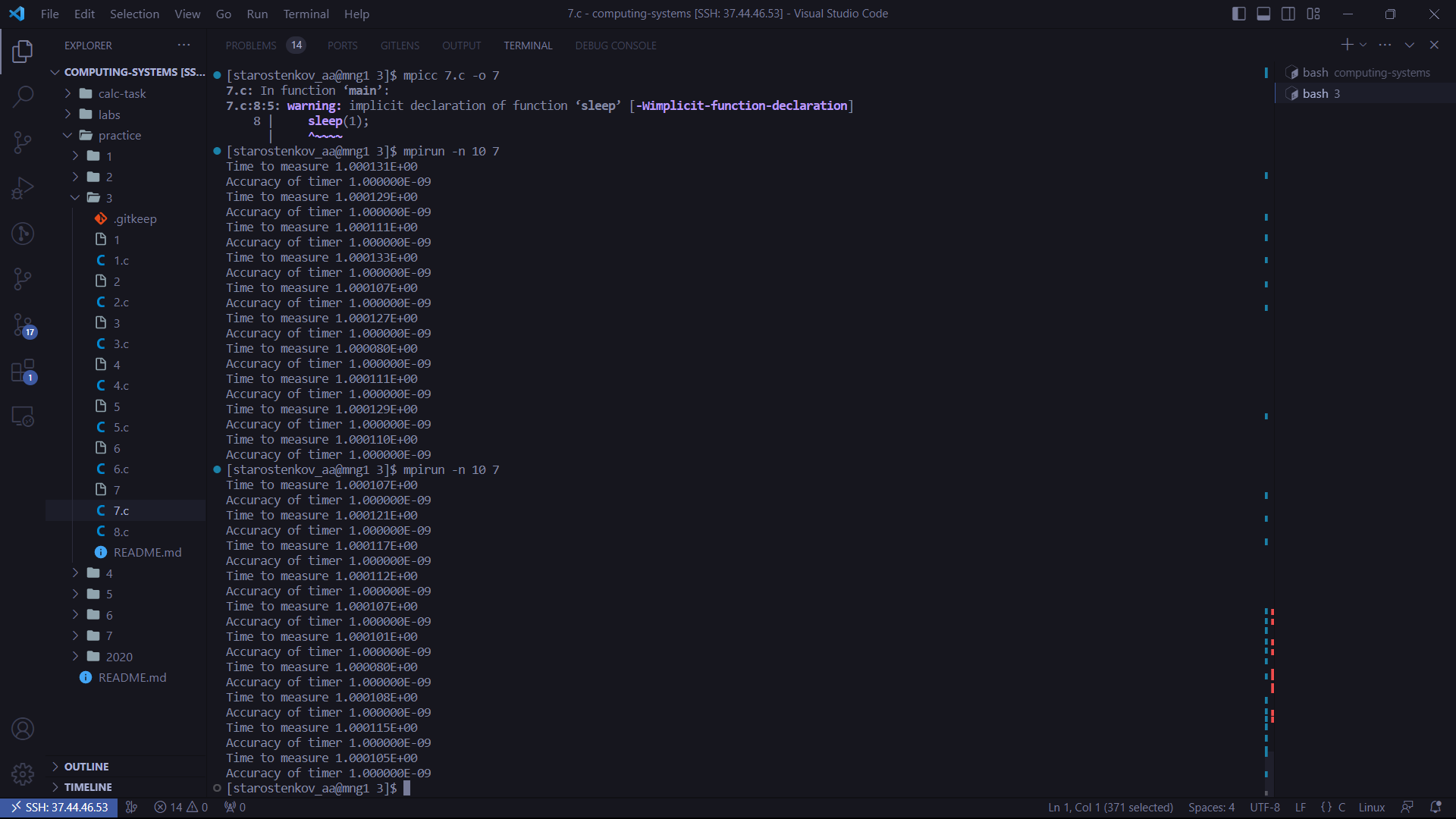


Рисунок 9 - Показания таймера

Вывод: программа работает корректно. Измеренное время слегка отличается от запуска к запуску. Это может происходить по целому ряду причин:

1. Linux является многопользовательской системой с вытесняющей многозадачностью – часть процессорного времени уходит на другие задачи.
2. Различное время доступа к памяти, в зависимости от загрузки шин и состояния кэшей процессора
3. Состояние конвейера процессора – время выполнения одних и тех же машинных команд может быть различным.

Точность таймера MPI составляет с, в то время как точность таймера OpenMP составляла с. Таким образом, таймер MPI на три порядка менее точен.

1. Программу 8 запустить с числом процессов, равным **10.**

Код



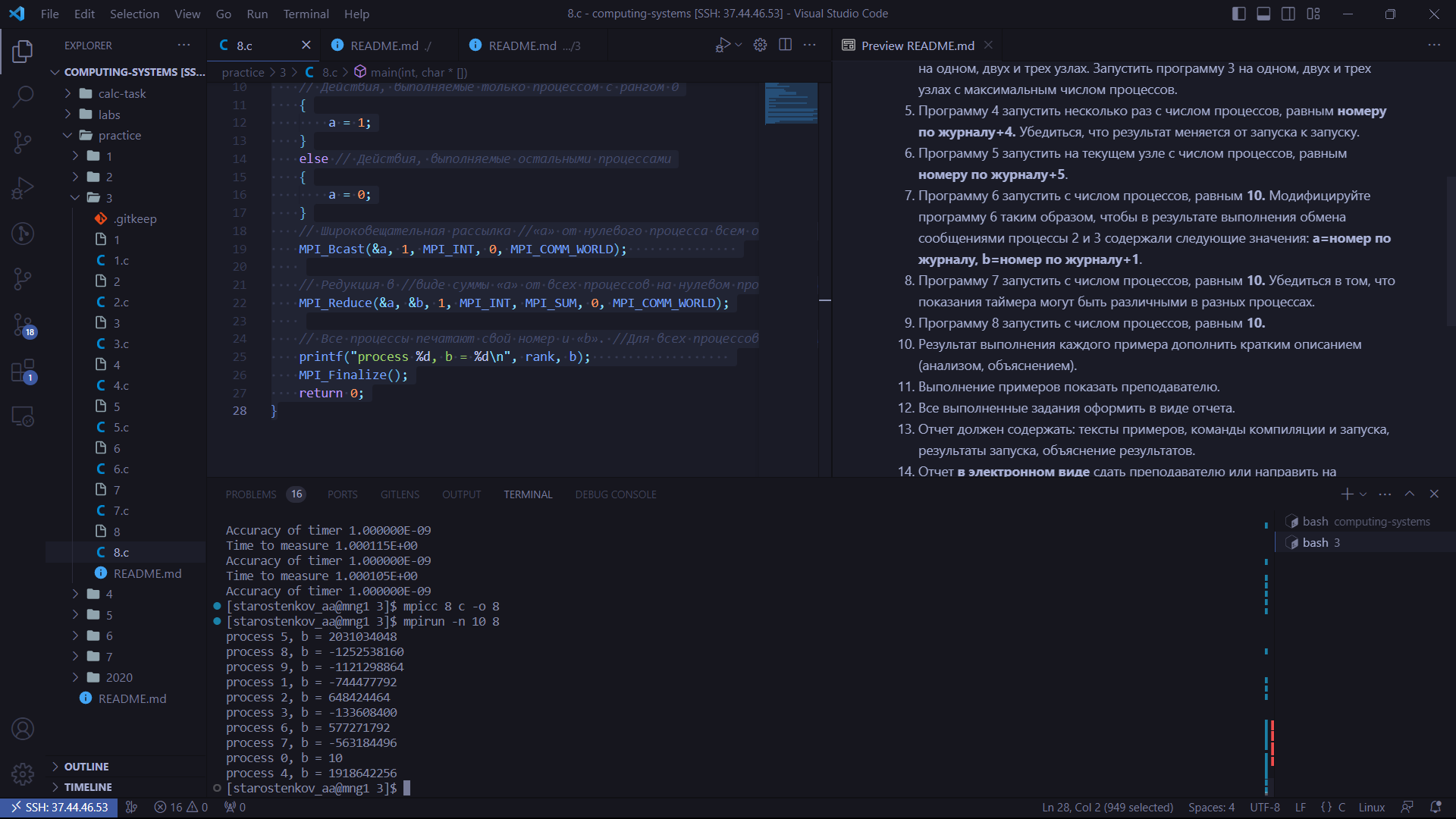


Рисунок 10 – Выполнение кода

Программа работает корректно. Каждый процесс определяет свой ранг, после чего действия в программе разделяются. Все процессы, кроме процесса с рангом 0, передают значение своего ранга нулевому процессу. Процесс с рангом 0 сначала печатает значение своего ранга, а далее последовательно принимает сообщения с рангами процессов и также печатает их значения. Порядок приема сообщений заранее не определен и зависит от условий выполнения параллельной программы (более того, этот порядок может изменяться от запуска к запуску).