Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

—

Институт компьютерных наук и кибербезопасности

**Высшая школа кибербезопасности**

**ЛАБОРАТОРНАЯ РАБОТА №3**

**СИНХРОНИЗАЦИЯ ПРОЦЕССОВ**

по дисциплине «Операционные системы»

Выполнил

студенты гр. 5131001/30002 Мишенев Н. С.

<*подпись*>

Руководитель

программист Огнёв Р. А.

<*подпись*>

Санкт-Петербург

2024г.

**СОДЕРЖАНИЕ**

[ЦЕЛЬ РАБОТЫ 3](#_Toc1590536092)

[ХОД РАБОТЫ 3](#_Toc32477232)

[1. Описание разработанного алгоритма и обоснование его эффективности. 4](#_Toc2094768874)

[2. Исходный код реализованных функций narrow\_bridge\_init(), arrive\_bridge() и exit\_bridge() с подробными комментариями. 4](#_Toc2047284055)

[> narrow\_bridge\_init(...) 4](#_Toc903408690)

[> arrive\_bridge(...) 5](#_Toc1136113854)

[> exit\_bridge(...) 5](#_Toc1842974884)

[3. Диаграмма взаимодействия процессов на «узком мосте». 5](#_Toc1115237777)

[4. Полученный в результате выполнения программы вывод. 6](#_Toc1883633193)

[ВЫВОД 8](#_Toc827901645)

# **ЦЕЛЬ РАБОТЫ**

Цель работы – изучение примитивов синхронизации и методов работы с ними, решение классической задачи узкого моста и тестирование решения в рамках операционной системы Pintos.

# **ХОД РАБОТЫ**

## **Описание разработанного алгоритма и обоснование его эффективности.**

В ходе выполнения лабораторной работы был разработан алгоритм, который позволяет решать классическую задачу узкого моста. Алгоритм действует следующим образом:

После появления первой машины, она запускается на мост, и, если вторая пришедшая машина едет в том же направлении она тоже запускается на мост, иначе ждет и после этого запускается на мост. Во время прохождения моста этими двумя машинами, к мосту подъезжают оставшиеся машины.

Как только на мосту оказывается 0 машин, функция ***exit\_bridge(),*** вызывает созданную функцию ***notify(),*** которая запускает на мост необходимые машины, с учетом их приоритета (обычная машина или машина скорой помощи).

При этом, если последняя машина ехала, например, слева направо, то машины будут запускаться справа налево, таким образом решается проблема голодания процессов.

## **Исходный код реализованных функций narrow\_bridge\_init(), arrive\_bridge() и exit\_bridge() с подробными комментариями.**

### **> narrow\_bridge\_init(...)**

Для решения задачи была создана система семафоров и несколько констант, позволяющих отслеживать состояние моста и машин в любой момент времени.

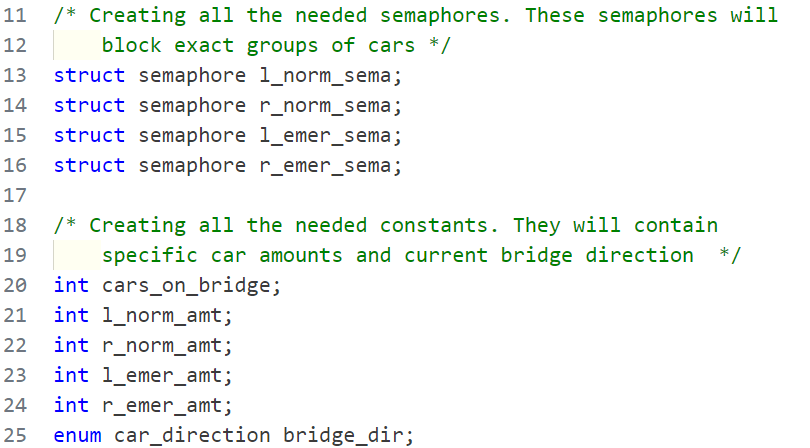


Рис. 1. Созданные константы и семафоры.

Далее все они были инициализированы своими значениями в функции ***narrow\_bridge\_init().***

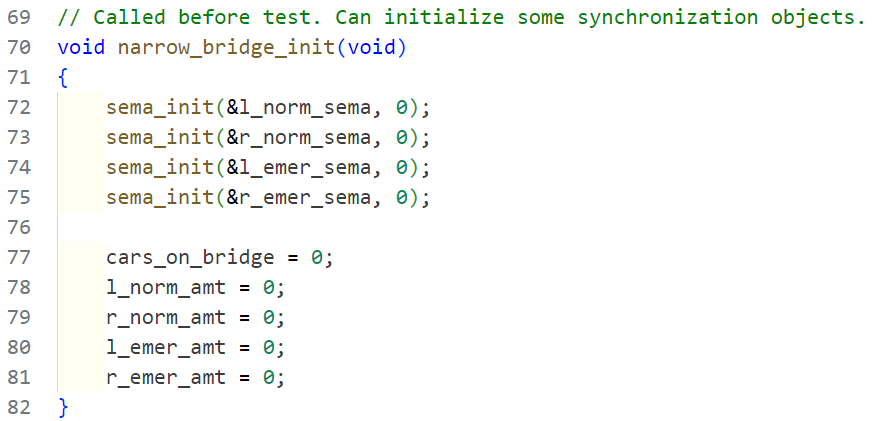


Рис. 2. Инициализация констант и семафоров.

### **> arrive\_bridge(...)**

При прибытии, первая и вторая машины рассматриваются отдельно, так как к этому моменту еще не приехали остальные, поэтому первая пришедшая машина устанавливает сторону движения на мосту, а вторая, если ее направление совпадает с первой, сразу же отправляется за ней. Далее, производится подсчёт пришедших машин и их последующая блокировка.

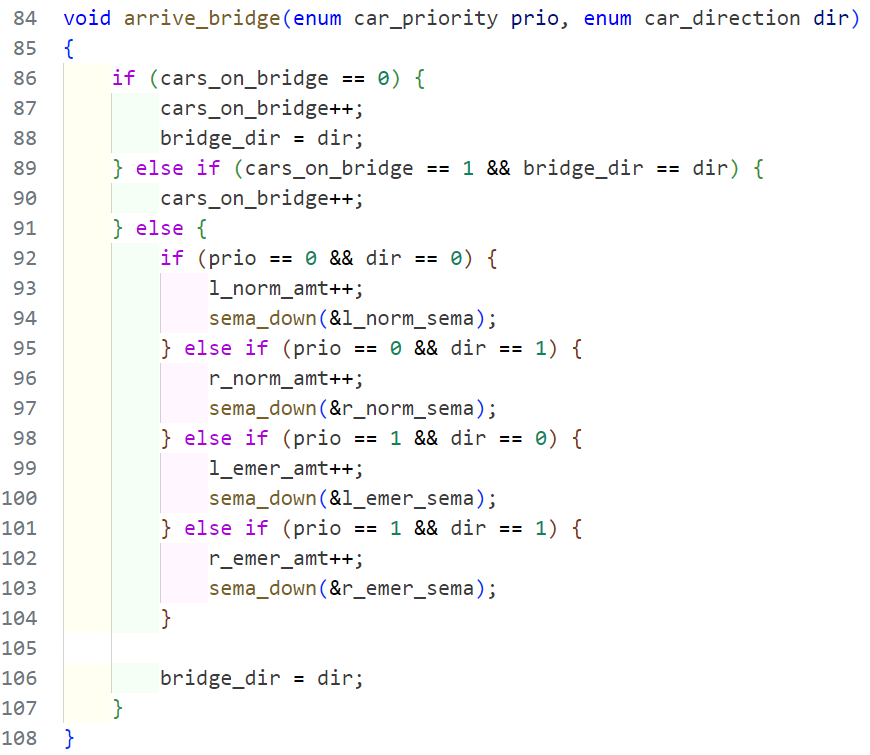


Рис. 3. Изменения в функции arrive\_bridge(...)

### **> exit\_bridge(...)**

В эту функцию было добавлено пробуждение машин. Когда машина является последней на мосту, и после её съезда мост полностью свободен, она сигнализирует следующим машинам, с учетом приоритета, на стороне прибытия, о том, что им можно проезжать, если таковые ожидают своей очереди.



Рис. 4. Изменения в функции exit\_bridge(...)

Это производится с помощью функции ***notify(...)*** аргументом которой является строка, обозначающая сторону и приоритет машины.

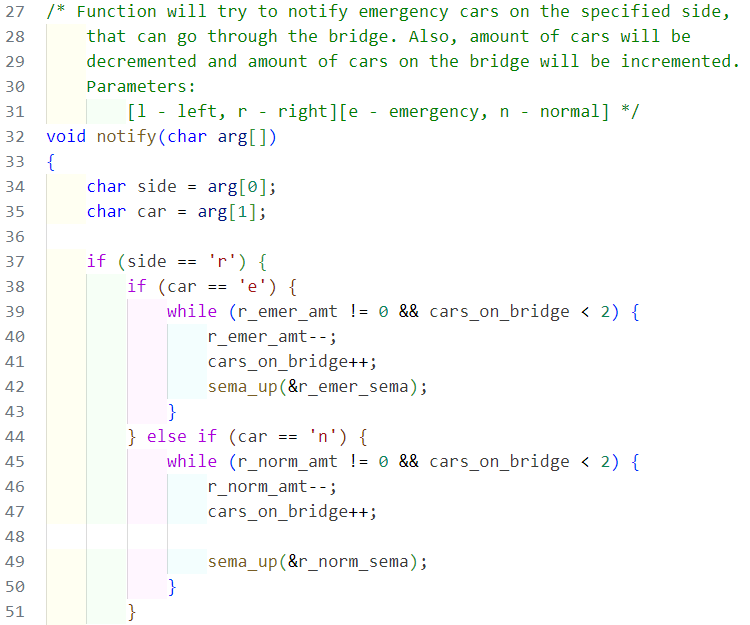


Рис. 5. Функция notify(...)

Функция декрементирует количество соответствующих ожидающих машин, инкрементирует количество машин на мосту и поднимает семафор, который ответственен за этот тип машин. При этом, учитывается общее количество машин на мосту.

## **Диаграмма взаимодействия процессов на «узком мосте».**

В соответствии с разработанным алгоритмом, была построена диаграмма взаимодействии “машин на мосту”, которая приведена ниже:

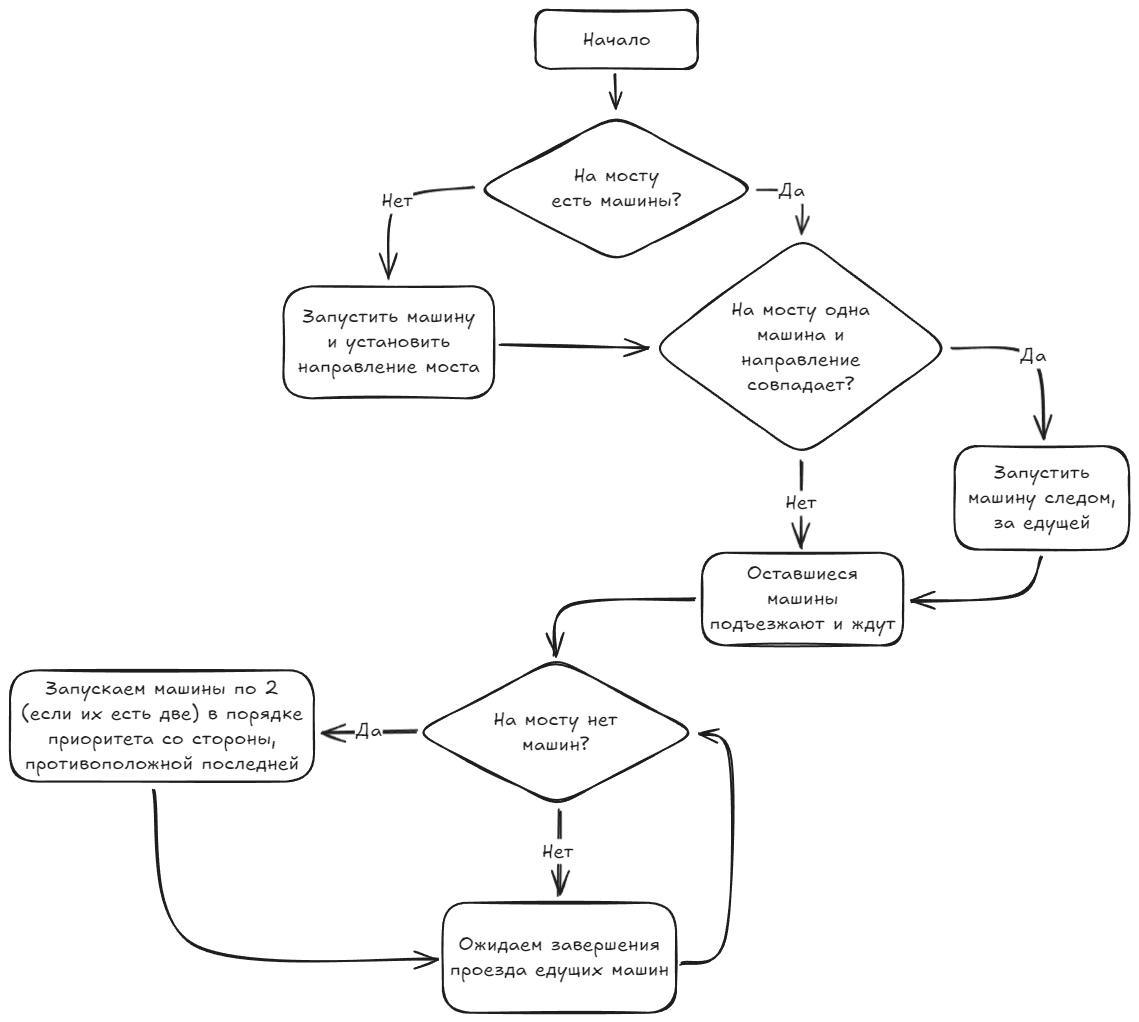


Рис. 6. диаграмма взаимодействии процессов

## **Полученный в результате выполнения программы вывод.**

После компиляции ядра с внедренным решением был проведен ряд тестов, результаты некоторых будут упомянуты в отчете.

Были рассмотрены ситуации, когда машин очень мало, демонстрирующие правильность работы учёта приоритета, ситуации, когда машины распределены равномерно и когда неравномерно. Результаты выполнения тестов приведены ниже.

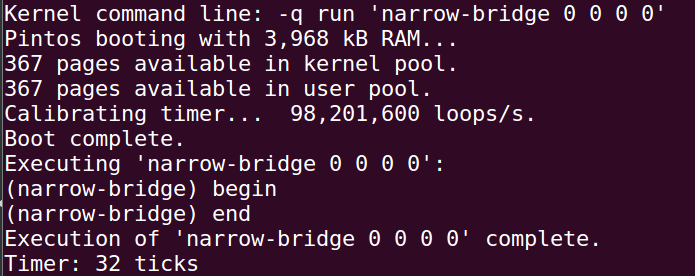


Рис. 7. Результаты теста “narrow-bridge 0 0 0 0”

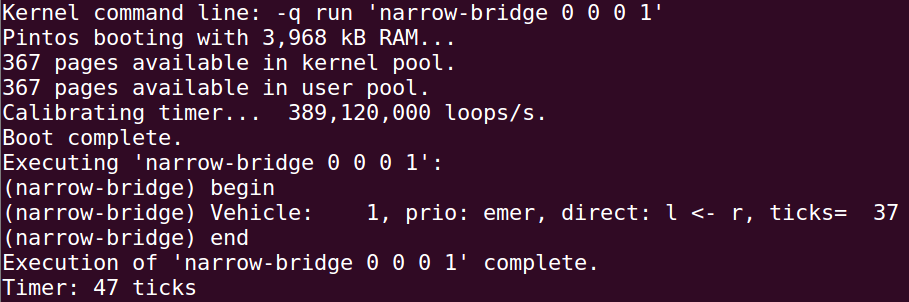


Рис. 8. Результаты теста “narrow-bridge 0 0 0 1”

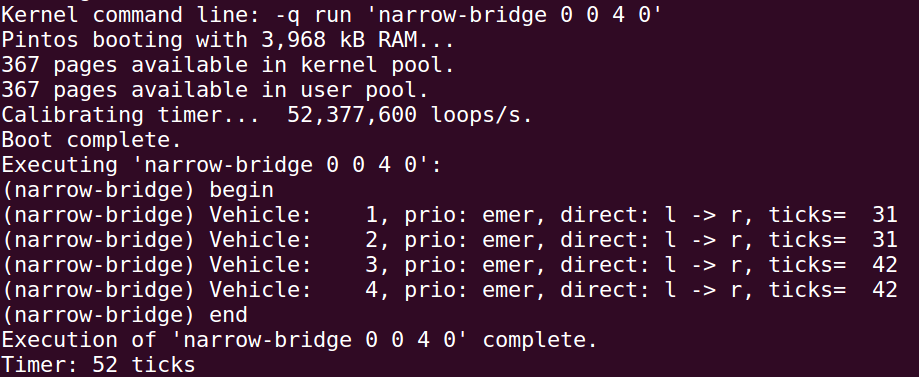


Рис. 9. Результаты теста “narrow-bridge 0 0 4 0”

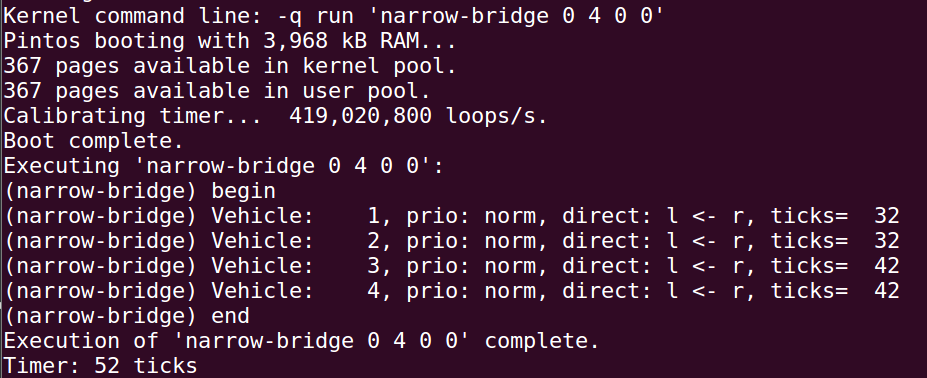


Рис. 10. Результаты теста “narrow-bridge 0 4 0 0”

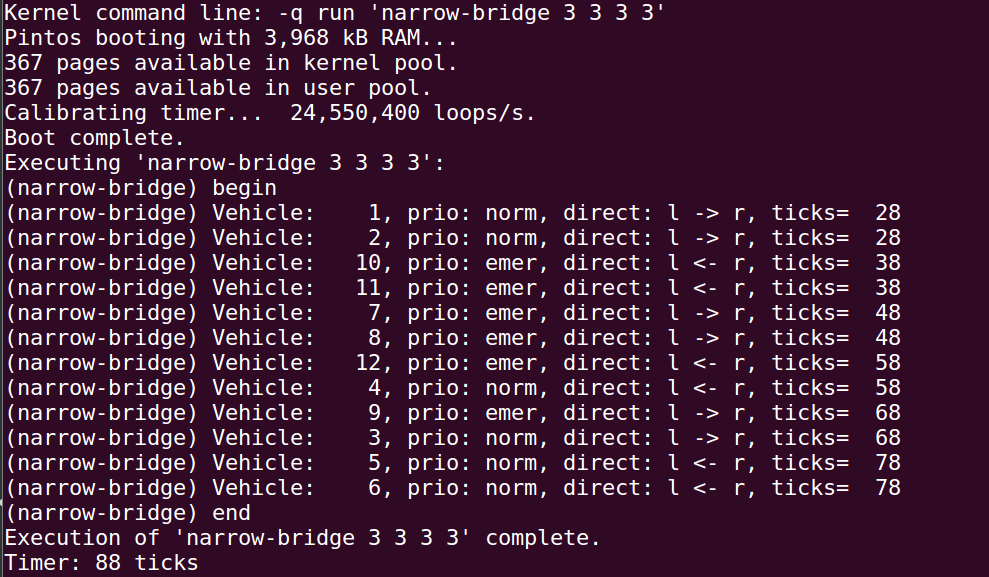


Рис. 11. Результаты теста “narrow-bridge 3 3 3 3”

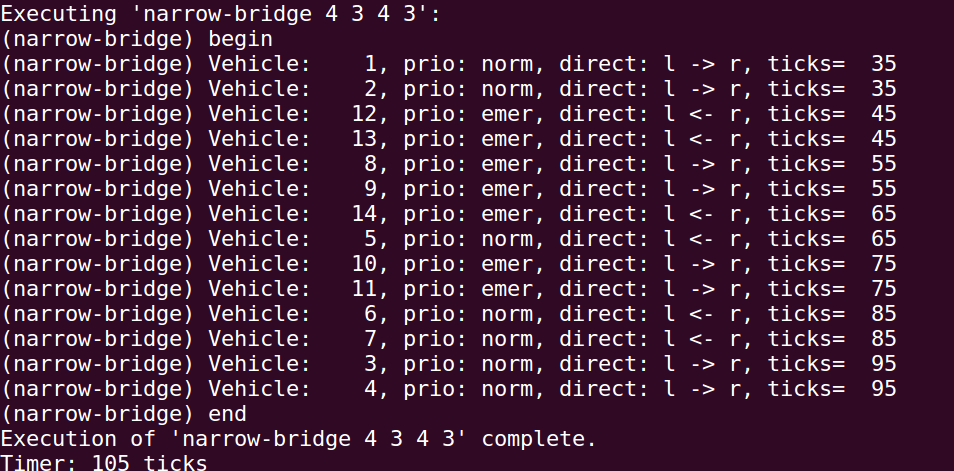


Рис. 12. Результаты теста “narrow-bridge 4 3 4 3”

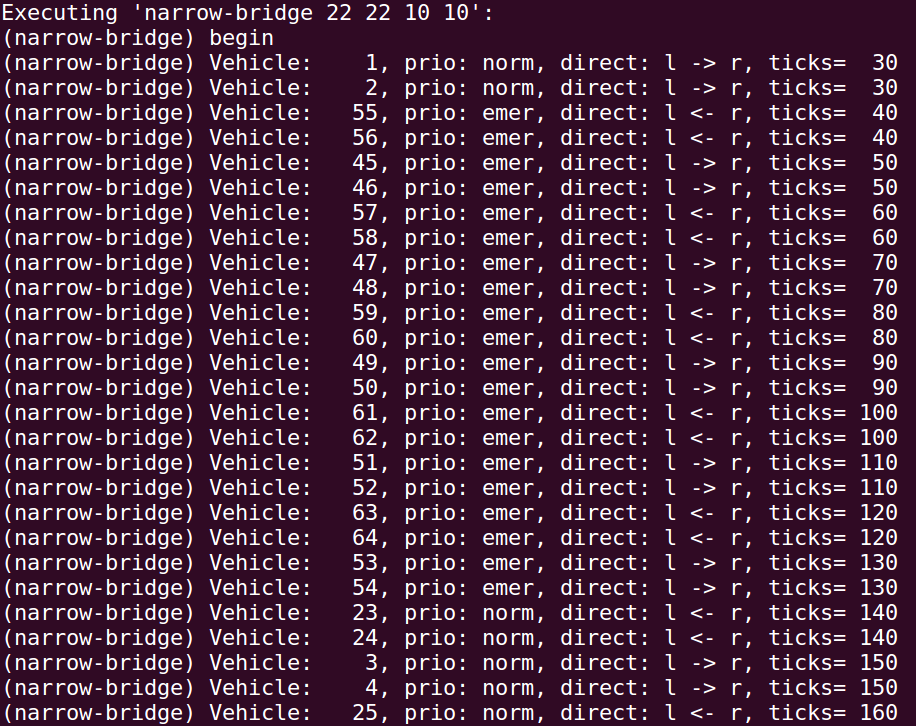


Рис. 13. Результаты теста “narrow-bridge 22 22 10 10”

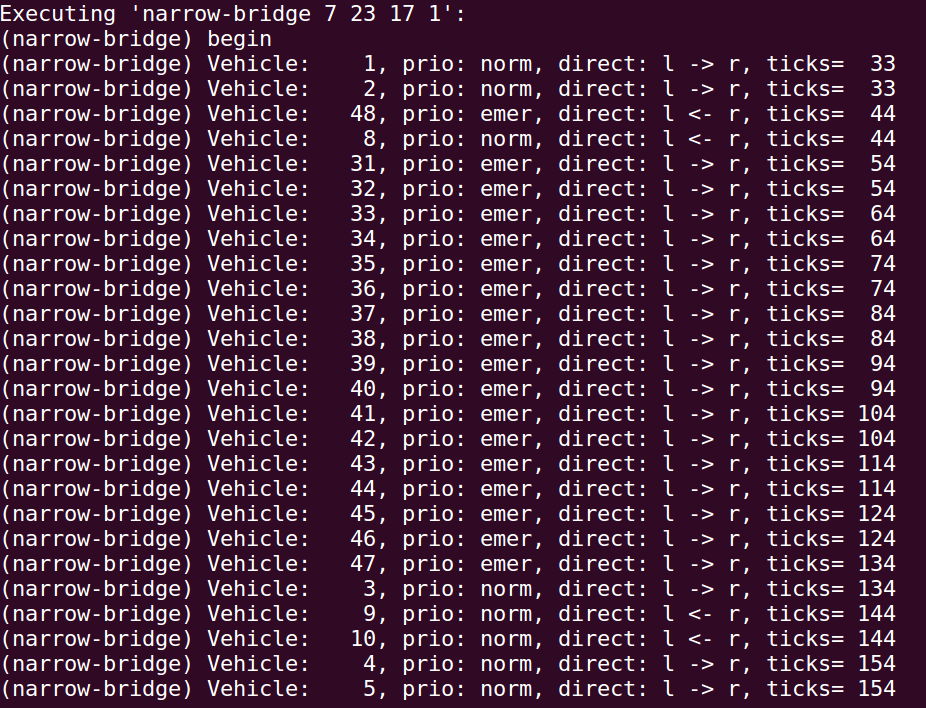


Рис. 14. Результаты теста “narrow-bridge 7 23 17 1”

# **ВЫВОД**

В ходе работы были изучены примитивы синхронизации и методы работы с ними, было разработано решение классической задачи узкого моста и проведено тестирование решения в рамках операционной системы Pintos.