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 №2**

**ПЛАНИРОВАНИЕ 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](#_Toc77368141)

[ХОД РАБОТЫ 3](#_Toc575575811)

[1. Диаграммы исполнения процессов. 4](#_Toc156144215)

[1. Исходный алгоритм. 4](#_Toc1555940231)

[2. Модифицированный алгоритм. 4](#_Toc369047296)

[2. Сравнительный анализ диаграмм. 4](#_Toc1840561648)

[3. Блок-схема нового разработанного алгоритма планирования. 5](#_Toc1831869020)

[4. Описание функций, которые применяются для организации планирования: их назначение, описание аргументов, возвращаемого значения, связей друг с другом. 5](#_Toc1379237252)

[5. Исходные коды системы планирования Pintos с внесенными модификациями и комментариями. 6](#_Toc571604465)

[> list.h + list.c 6](#_Toc1462806776)

[> Thread.h 8](#_Toc1137879097)

[> Thread.c 8](#_Toc2146160032)

[> Synch.h 10](#_Toc845086476)

[> Synch.c 10](#_Toc980648277)

[6. Описание тестовых задач: имя теста и описание действий, которые в нем совершаются; полученные при запуске результаты, анализ полученных результатов. 11](#_Toc1671247326)

[ВЫВОД 12](#_Toc1054813592)

**ЦЕЛЬ РАБОТЫ**

Цель работы – изучение механизмов планирования процессов, разработка алгоритма приоритетного планирования и внедрение разработанного алгоритма в учебную операционную систему Pintos.

ХОД РАБОТЫ

## **Диаграммы исполнения процессов.**

### **Исходный алгоритм.**

Исходный алгоритм планирования процессов в ОС Pintos, не учитывал приоритет процессов. Диаграмма исполнения процессов была составлена для 4-го варианта задания.

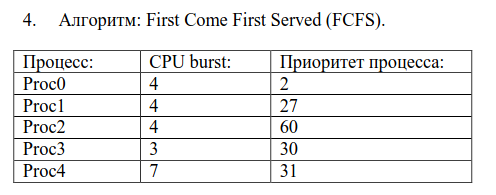


Рис. 1. Вариант задания.

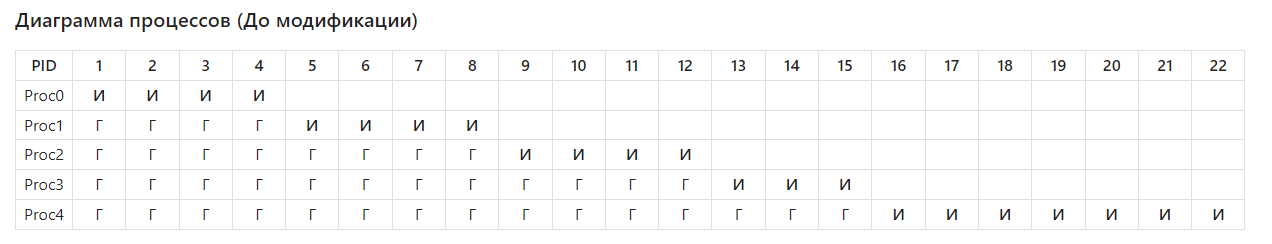
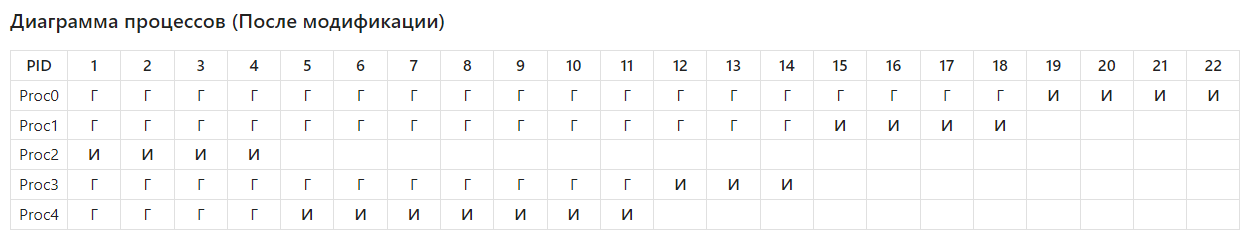


Рис. 2. Диаграмма исполнения процессов.

### **Модифицированный алгоритм.**

После модификации алгоритма планировщик выбирает из списка готовых процессов, процесс с наивысшим приоритетом, поэтому вид диаграммы изменился:



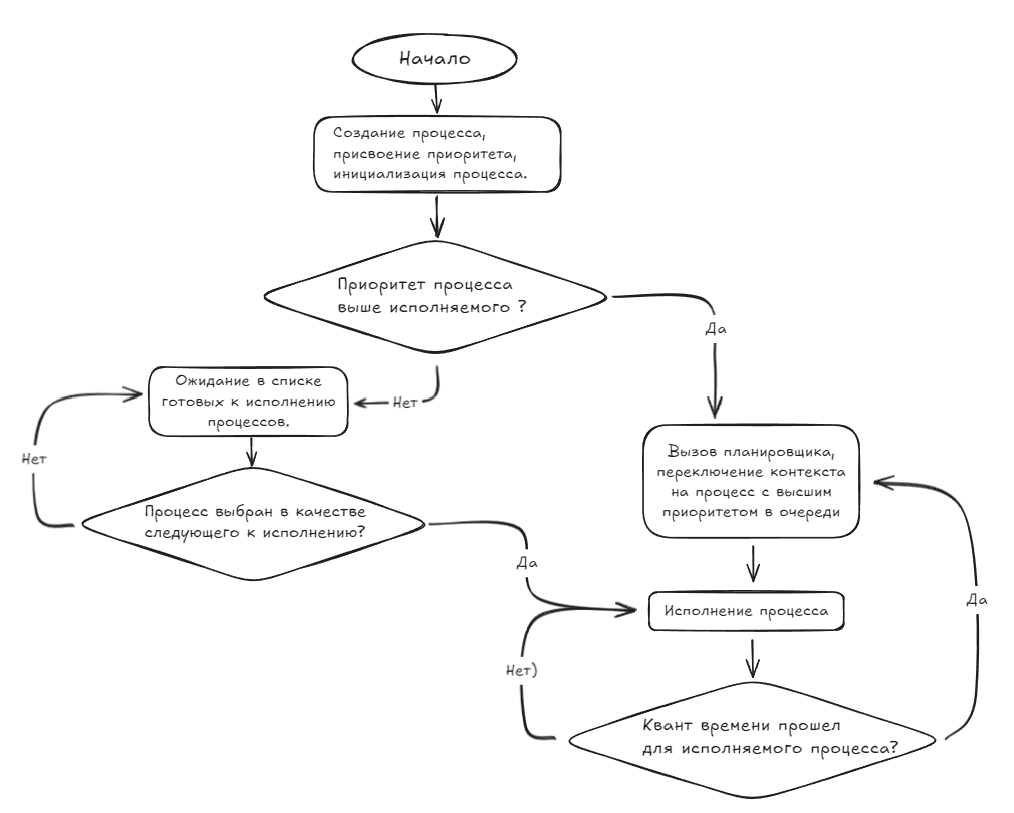
*Рис. 3. Новая диаграмма исполнения процессов.*

## **Сравнительный анализ диаграмм.**

Как можно заметить, из-за изменения алгоритма планирования изменился порядок исполнения процессов. Процессы с высшим приоритетом исполняются раньше, чем процессы с низшим приоритетом.

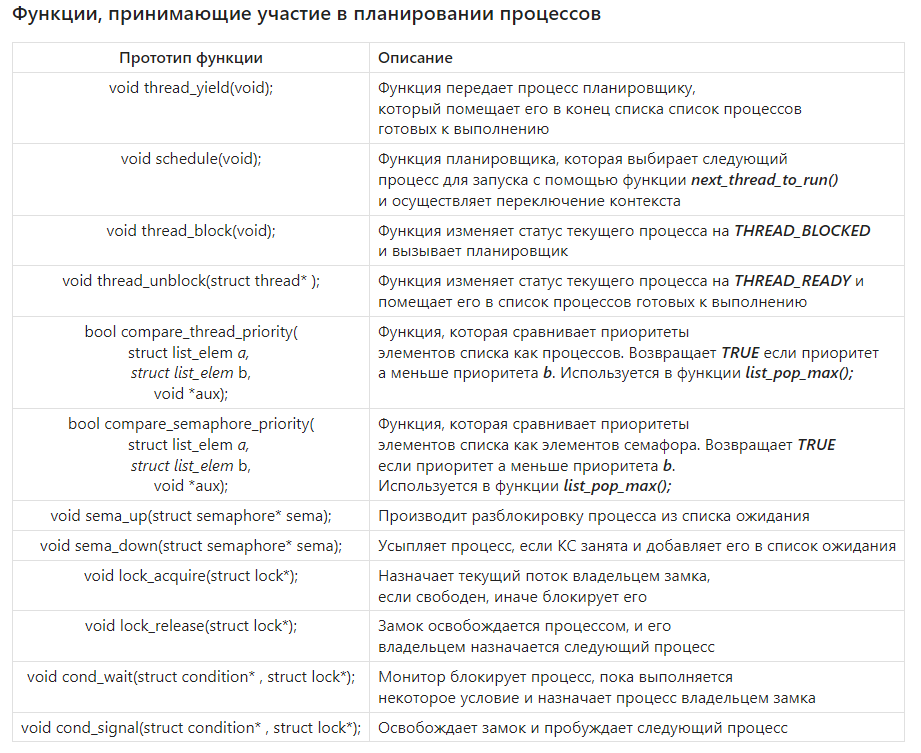
## **Блок-схема нового разработанного алгоритма планирования.**

Для нового алгоритма планирования была создана соответствующая ему блок-схема:

*Рис. 4. Блок-схема для нового алгоритма планирования.*

## **Описание функций, которые применяются для организации планирования: их назначение, описание аргументов, возвращаемого значения, связей друг с другом.**

Были рассмотрены файлы ***thread.c*** и ***synch.c***, и функции отвечающие за организацию планирования в них. Далее была составлена таблица, которая также включает в себя функции, разработанные в процессе выполнения лабораторной работы.



*Таблица 1. Функции, принимающие участие в планировании процессов.*

## **Исходные коды системы планирования Pintos с внесенными модификациями и комментариями.**

Для реализации очереди процессов в ОС Pintos используется описанный в самой ОС двусвязный список, поэтому в реализацию списка были внесены новые функции, позволяющие отыскивать и удалять максимальный элемент, а также находить определенный элемент.

Рассмотрим эти изменения.

### **> list.h + list.c**

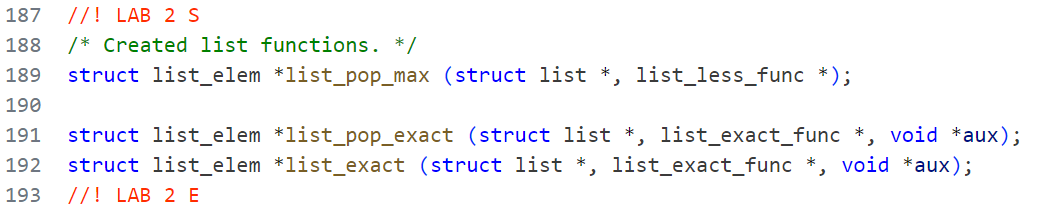


Рис. 5. Прототипы созданных функций в файле list.h

В заголовочном файле были описаны прототипы созданных функций, рассмотрим их назначение и реализацию:

***list\_pop\_max(...)*** - находит и удаляет из списка максимальный элемент, определенный переданной функцией.

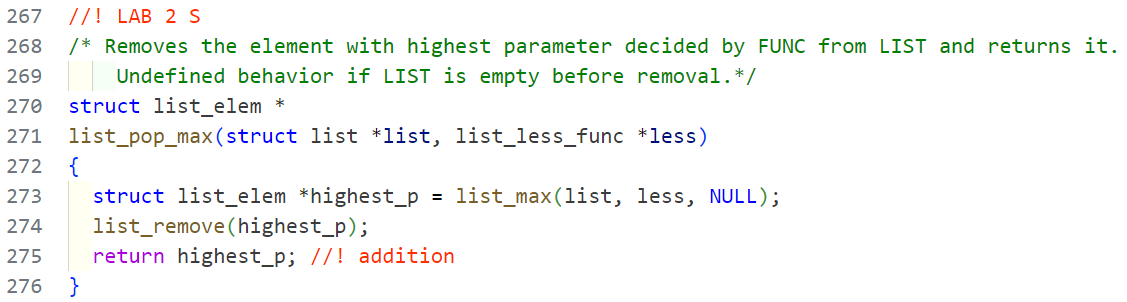
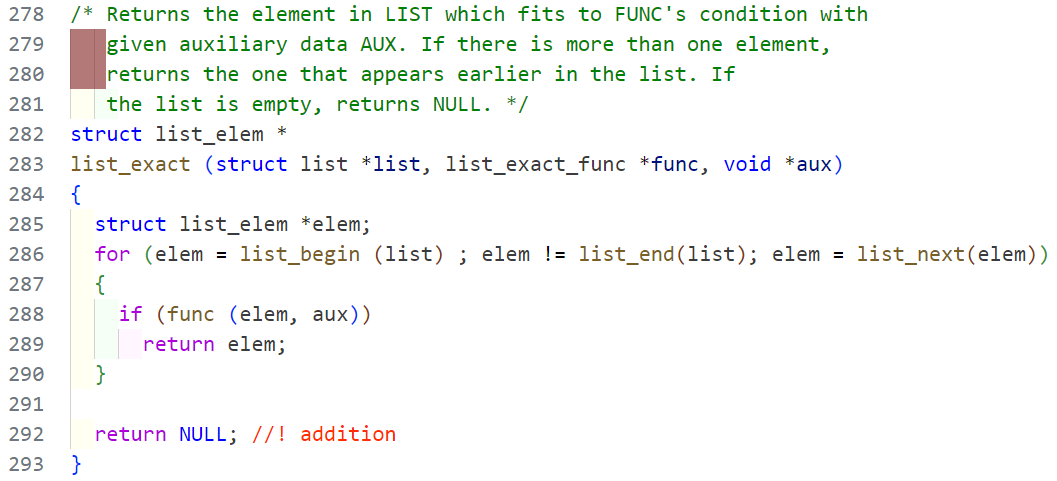


Рис. 6. Реализация функции list\_pop\_max() в файле list.c

***list\_exact(...)*** - возвращает первый элемент списка, который подходит под условие переданной функции.

Рис. 7. Реализация функции list\_exact() в файле list.c

***list\_pop\_exact(...)*** - удаляет из списка первый элемент, который подходит под условие переданной функции.

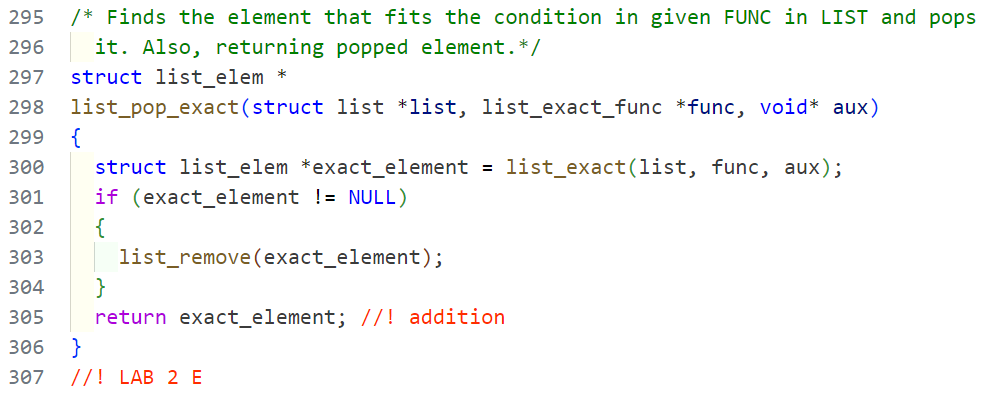
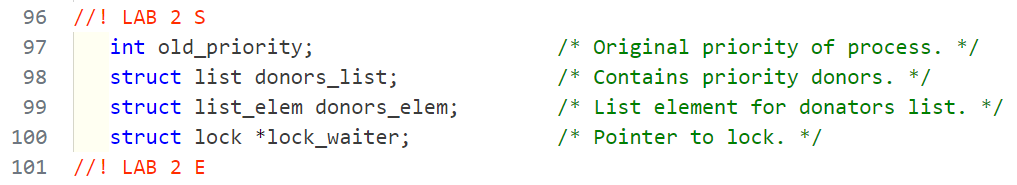


Рис. 7. Реализация функции list\_pop\_exact() в файле list.c

### **> Thread.h**

В структуру процесса были внесены следующие изменения:



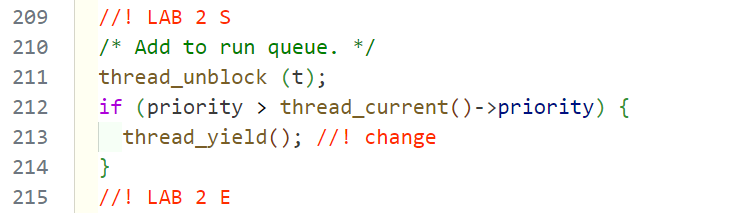
*Рис. 5. Изменения в структуре процесса.*

Были добавлены поля для сохранения изначального приоритета процесса, для сохранения списка “доноров” приоритета, а также был добавлен указатель на замок, который ожидает данный процесс.

Также в этот файл был добавлен прототип функции ***compare\_thread\_priority(...)***, которая была описана в таблице выше.

### **> Thread.c**

В функцию ***thread\_create(..)*** было добавлено учитывание приоритета процесса при его последующей планировке:



*Рис. 6. Изменения в функции thread\_create().*

Была изменена функция ***thread\_set\_priority()*** так, чтобы она учитывала старый приоритет процесса, который записан в ***old\_priority*** поле процесса.

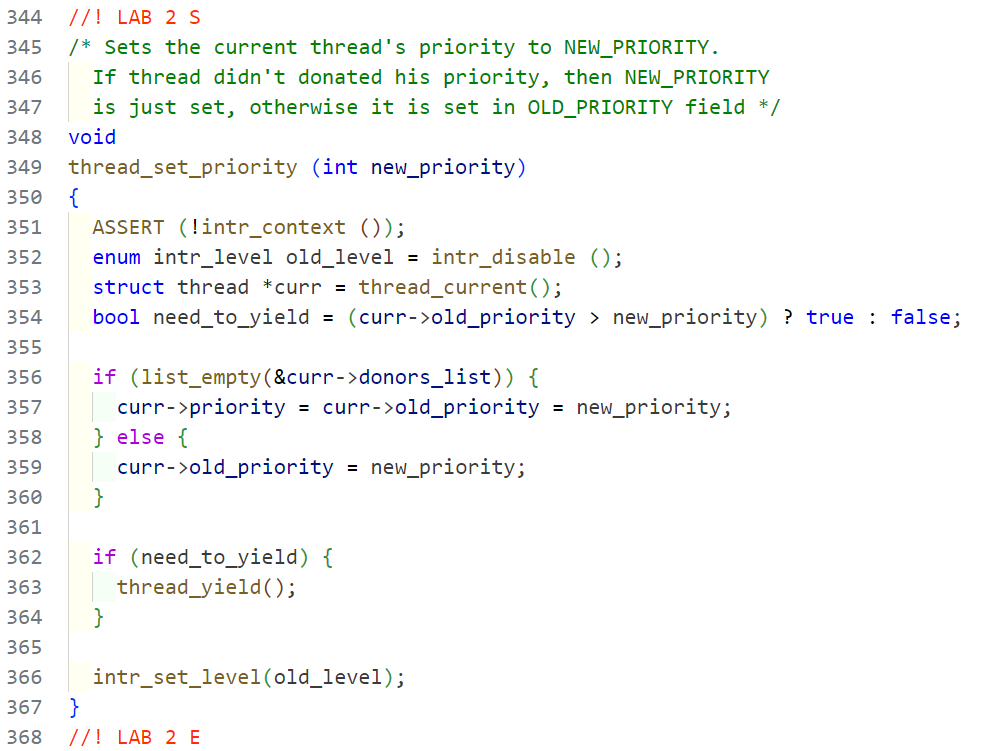


Рис. 7. Изменения в функции thread\_set\_priority().

Изменения затронули и функцию ***init\_thread(),*** в которую была добавлена инициализация добавленных ранее новых полей:

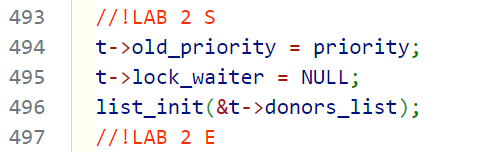


Рис. 7. Изменения в функции init\_thread().

Изменена функция планировщика ***next\_thread\_to\_run()*** таким образом, чтобы она выбирала процесс с наивысшим приоритетом из списка процессов готовых к исполнению. Правильный выбор осуществляется с помощью определенной ранее функции ***list\_pop\_max()***:

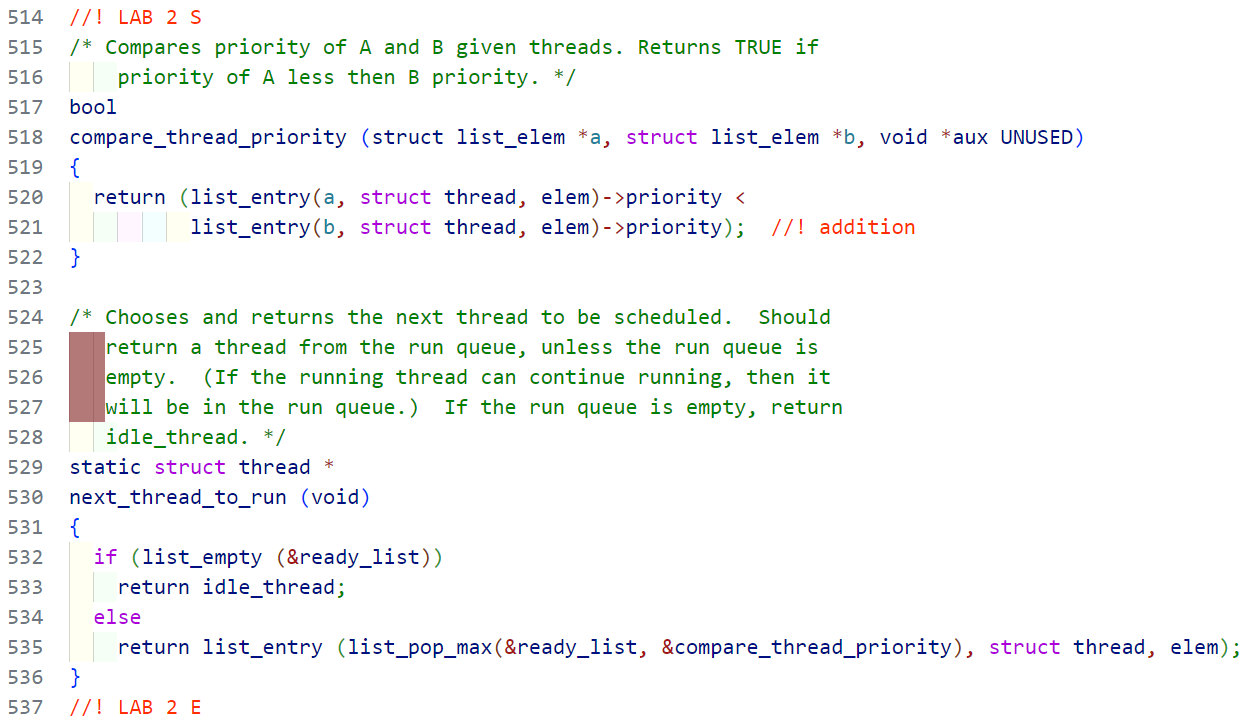


Рис. 8. Изменения в функции next\_thread\_to\_run();

### **> Synch.h**

В заголовочный файл был добавлен только прототип функции ***compare\_semaphore\_priority()***, которая также была описана в таблице выше.

### **> Synch.c**

Были внесены изменения в функцию “V” семафора, для того чтобы пробуждать процесс из списка ждущих с наивысшим приоритетом, а также учитывать его приоритет при планировании:

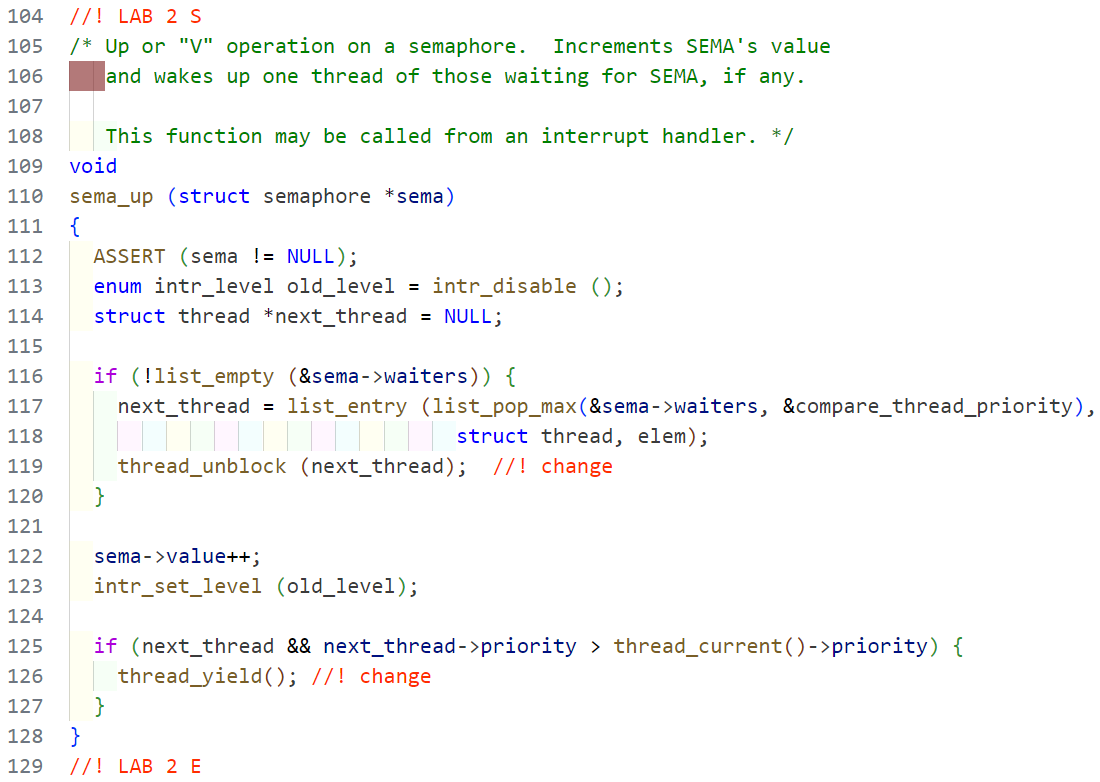


Рис. 9. Изменения в функции семафора sema\_up().

Для создания механизма распределения приоритета было разработано несколько функций. Рассмотрим назначение этих функций и их реализации:

***void priority\_donate(...) -*** функция, которая осуществляет передачу приоритета, если целевой процесс в этом нуждается, а также добавляет указатель на отправителя приоритета в список доноров целевого процесса.

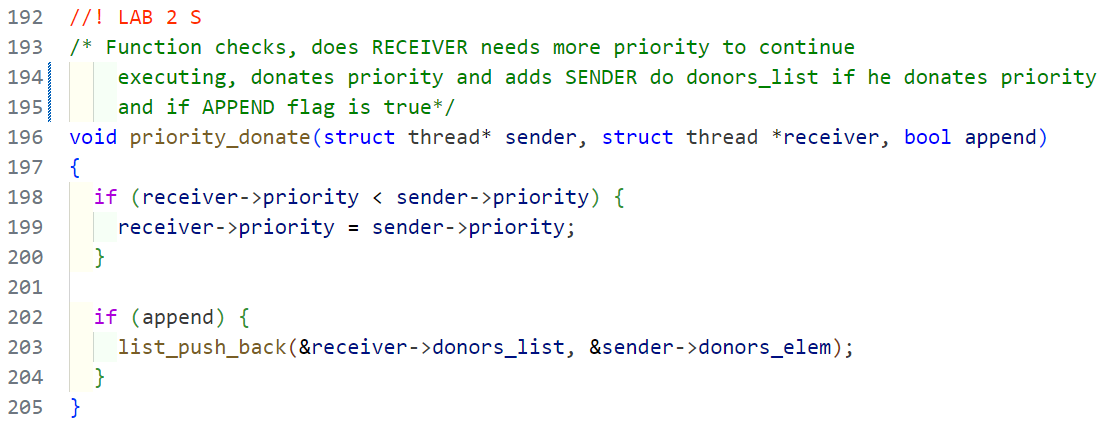


Рис. 10. Реализация функции priority\_donate().

***void find\_lock(...)*** *-* функция-фильтр, которая используется при итерации по списку с помощью функции ***find\_exact(...)***описанной выше. Возвращает элемент, указатель на замок, в котором совпадает с переданным в функцию.

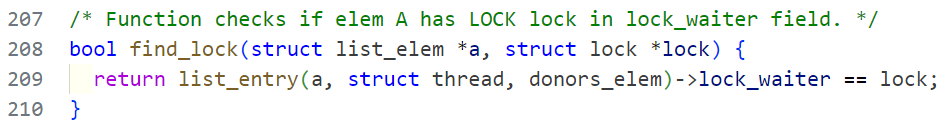


Рис. 11. Реализация функции find\_lock().

***void revert\_priority\_donation(...) -*** функция, которая находит процесс донор в списке доноров, удаляет указатель на него из списка и возвращает процессу донору свой изначальный приоритет.

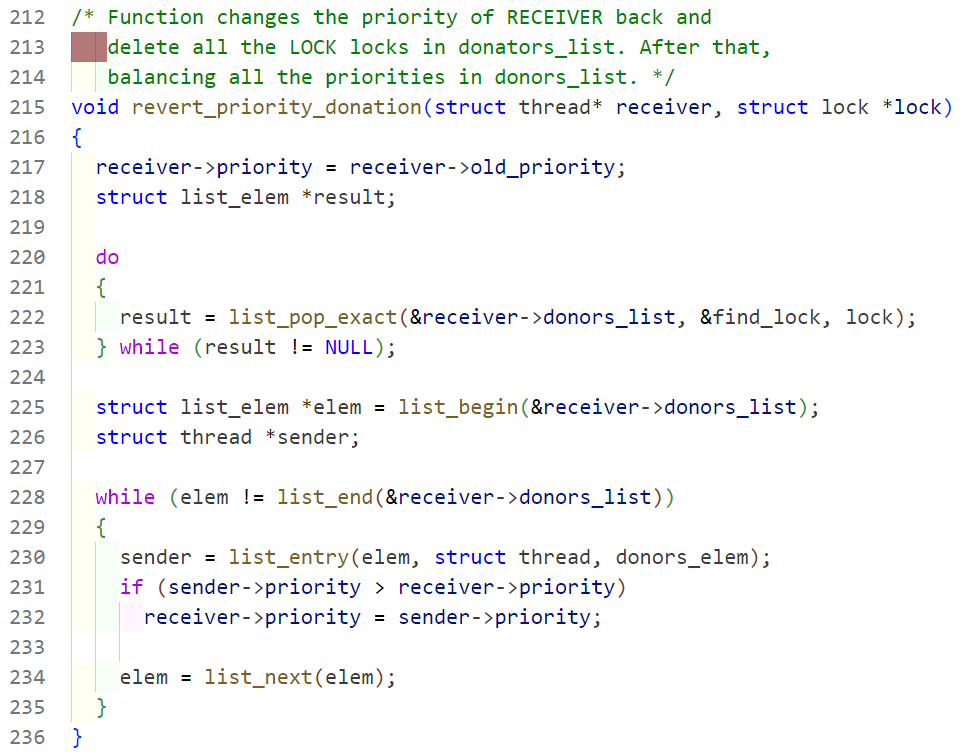


Рис. 12. Реализация функции revert\_priority\_donate().

Для того чтобы использовать систему пожертвования приоритетов, были внесены изменения в функции ***lock\_acquire()*** и ***lock\_release().*** В функции ***lock\_acquire()*** происходит пожертвование приоритета процессу, занимающему критическую секцию до тех пор, пока он не освободит замок.

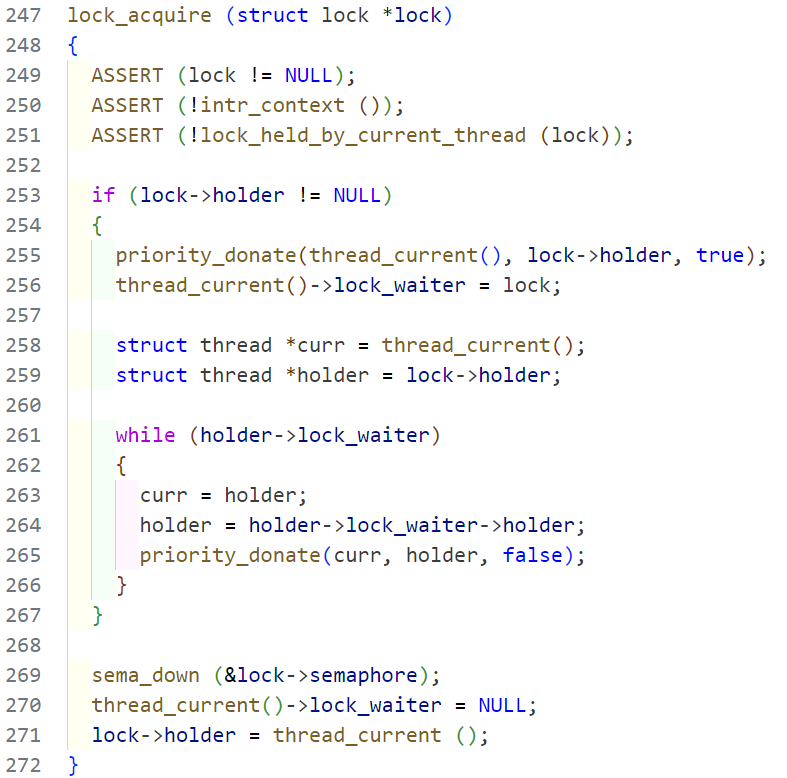


Рис. 13. Изменения в функции lock\_acquire().

Так же в функцию ***lock\_release()*** был добавлен вызов функции, отменяющей пожертвование приоритета, принцип работы которой был описан выше.

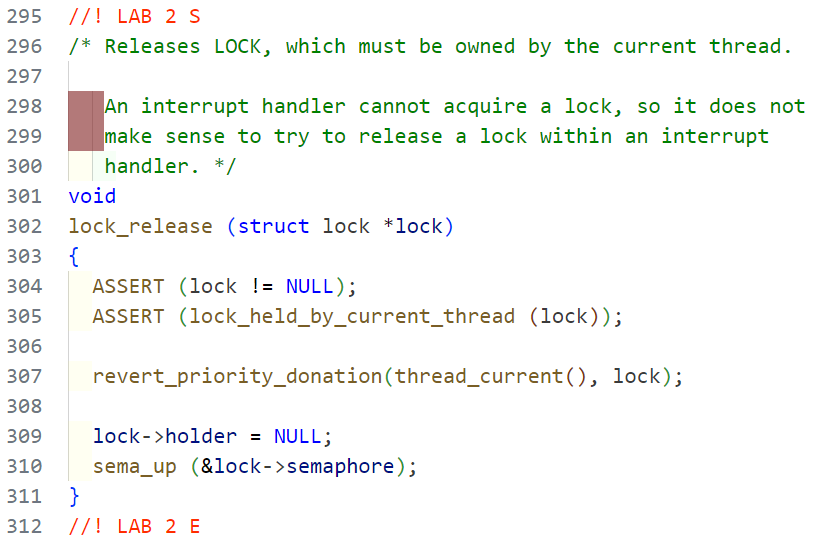


Рис. 14. Изменения в функции lock\_release().

Так же были внесены изменения в реализацию монитора. В описание элемента семафора было добавлено поле с приоритетом, которое, при инициализации монитора, устанавливалось равным приоритету процесса.

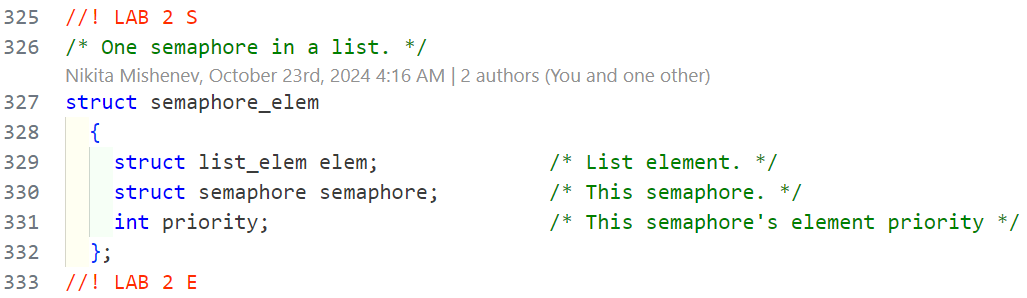


Рис. 15. Изменения в описании элемента semaphore\_elem.

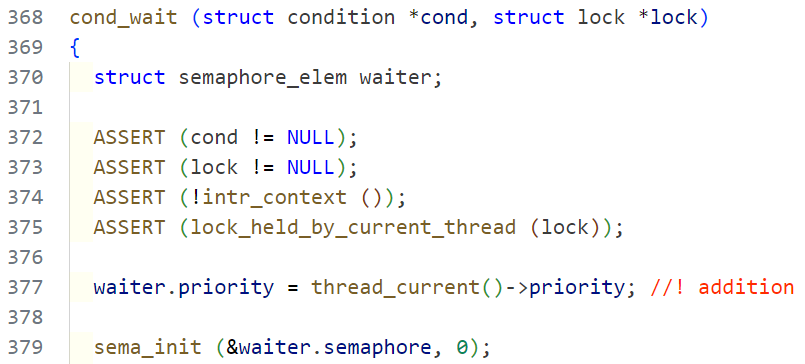


Рис. 16. Изменения в функции cond\_wait().

Изменения были также внесены и в функцию ***cond\_wait()***, такие, чтобы пробуждался семафор с наивысшим приоритетом процесса, ожидающего его разблокировку.

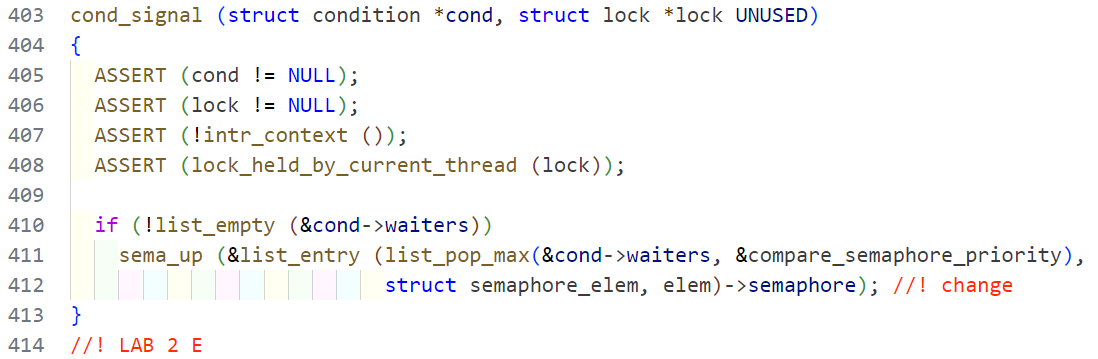


Рис. 17. Изменения функции cond\_signal()

## **Описание тестовых задач: имя теста и описание действий, которые в нем совершаются; полученные при запуске результаты, анализ полученных результатов.**

**alarm-priority** - проверка на корректность порядка пробуждение процессов с учетом приоритетов.

**priority-change** - проверка корректности порядка пробуждения процессов при изменении приоритета.

**priority-fifo** - проверка на корректность используемого алгоритма планирования (Round Robin).

**priority-preempt -** проверка на то, что поток с большим приоритетом выполняется в приоритетном режиме.

**priority-sema -** проверка на корректность порядка пробуждения процессов на семафоре.

**priority-condvar -** проверка на корректность порядка пробуждения процессов на мониторе.

**priority-donate-one -** проверка на корректность порядка передачи замка в соответствии с приоритетами созданных основным потоком процессов

**priority-donate-lower -** проверка на корректность последовательности действий. Понижение приоритета основного процесса не должно иметь эффект до завершения передачи приоритета.

**priority-donate-multiple -** проверка на корректность последовательности передачи приоритетов. Основной процесс должен в правильном порядке вернуть приоритеты процессам-донорам.

**priority-donate-multiple2 -** отличается от предыдущего теста, порядком возврата приоритетов.

**priority-donate-sema -** проверка механизма пожертвования проиритетов на семафоре.

**priority-donate-nest -** проверка множественного глубокого наследования на семафорах и замках

**priority-donate-chain -** проверка сохранения приоритетов при многократном последовательном пожертвовании, вырождающемся в “цепь”

При запуске тестов локально на виртуальной машине, все тесты проходятся успешно.

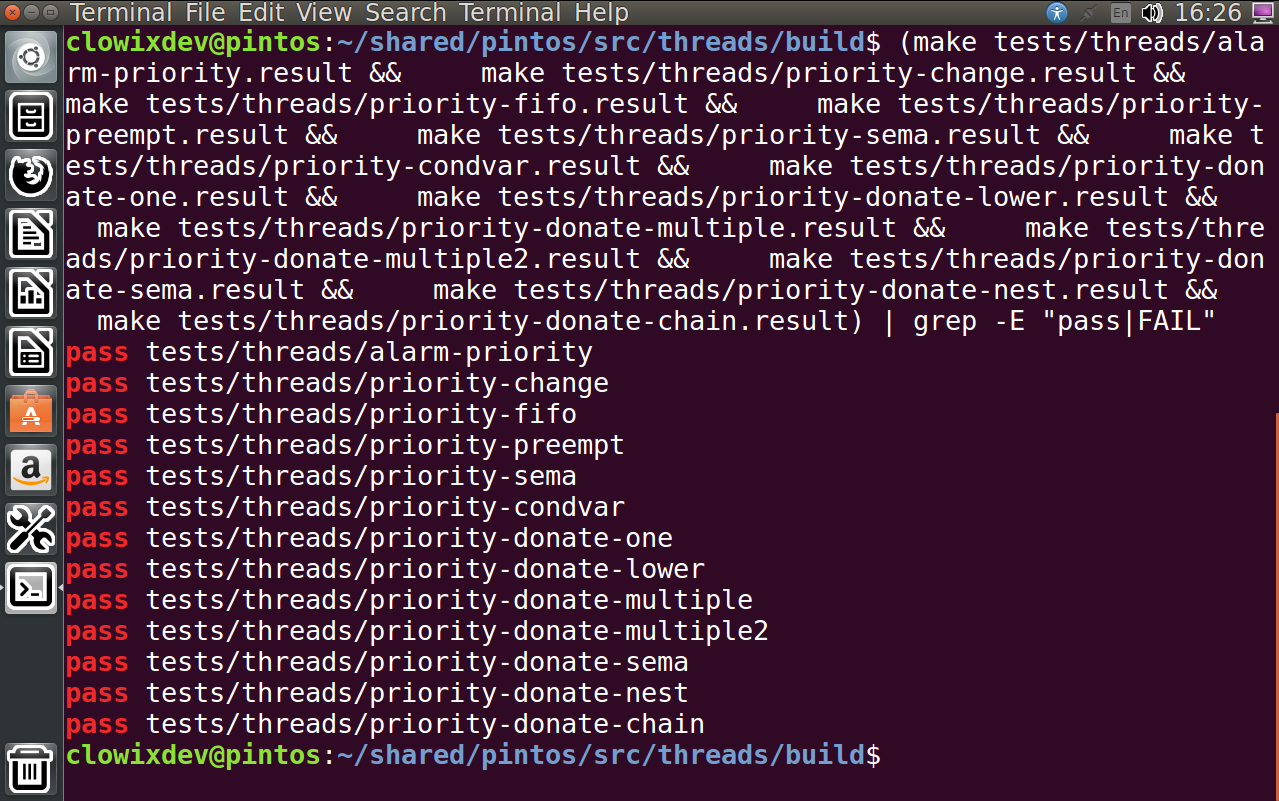


Рис. 18. Результаты запуска необходимых тестов.

**ВЫВОД**

В ходе работы был изучен механизм планирования процессов, также был разработан алгоритм приоритетного планирования и внедрён в учебную операционную систему Pintos.