**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

по дисциплине «Операционные системы»

на тему: «Синхронизация процессов»

Выполнил: студент гр. ИТП-11

Дроздов Д.С.

Принял: преподаватель стажёр

Карась О.В.

Гомель 2022

**Цель:** Изучить основные алгоритмы планирования процессов. Научиться обрабатывать несколько процессов, и синхронизировать их.

**Задача:**

1. Выполнить алгоритм синхронизации двух процессов *(Р0, Р1)* «*переменная–замок*», использующих общие ресурсы, для данных приведенных в таблице 2.1. Алгоритм планирования процессов *Round Robin**(*RR*)*, величина кванта времени 3. Результаты оформить в виде таблицы иллюстрирующей работу процессов.

2.Выполнить алгоритм синхронизации двух процессов (*Р0, Р1*) «*строгое – чередование*», использующих общие ресурсы, для данных приведенных в таблице 2.1. Алгоритм планирования процессов *Round Robin**(RR)*, величина кванта времени 3. Результаты оформить в виде таблицы иллюстрирующей работу процессов.

3. Выполнить алгоритмы синхронизации процессов (*Р0, Р1*) «*переменная – замок*» и «*строгое – чередование*», использующих общие ресурсы, при наличии третьего процесса (*Р2*), не использующего ресурсы процессов *Р0*, *Р1*. Данные процессов (Р0, Р1) «приведенных в таблице 2.1, процесс *Р2* появляется каждый 6 квант времени, длительность процесса равна 3квантам. Алгоритм планирования процессов *Round Robin (RR)*, величина кванта времени 3. Если процесс *Р2* выполниться не успел, новый его экземпляр в очередь не ставится. Процесс Р2 не может прервать выполнение критической секции. Результаты оформить в виде таблиц иллюстрирующих работу процессов.

4. Выполнить алгоритм синхронизации четырёх процессов *(Р0, Р1, Р2, Р3)* «*алгоритм булочной*», использующих общие ресурсы. Процессы выбираются из таблицы 2.1, согласно таблице 2.2. При каждой постановке в очередь критической секции, вычисляется номер присваиваемый процессу. Алгоритм планирования процессов *Round Robin (RR)*, величина кванта времени 3. Результаты оформить в виде таблицы иллюстрирующей работу процессов, в таблице указывать номер.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PO | Время возникновения | 1 | 4 | 10 | 16 | 21 | 29 |
| Продолжительность | 2 | 1 | 1 | 1 | 1 | 1 |
| P1 | Время возникновения | 8 | 12 | 15 | 18 | 22 | 26 |
| Продолжительность | 1 | 1 | 1 | 1 | 2 | 1 |

**1. Алгоритм взаимодействия двух процессов «Переменная – замок»**

Суть алгоритма проста: когда возникает вход в критическую секцию, мы ждем, пока эта критическая секция выполнится, затем выполняем сам процесс, при этом мы должны учитывать, что нам дается 3 кванта времени на одну итерацию. Когда один процесс выполняется, а другой в это время входит в критическую секцию, то он будет ожидать, а потом только войдет в нее.

На рисунке 1 представлен результат выполнения 1 задания.



Рисунок 1 - Алгоритм «Переменная – замок» для P0 и P1

**2. Алгоритм взаимодействия двух процессов «Строгое чередование»**

Очередной подход будет использовать общую переменную с начальным значением 0. Только теперь она будет играть не роль замка для критического участка, а явно указывать, кто может следующим войти в него. Пример выполнения второго задания указан на рисунке 2.

На рисунке 2 представлен результат выполнения 2 задания.

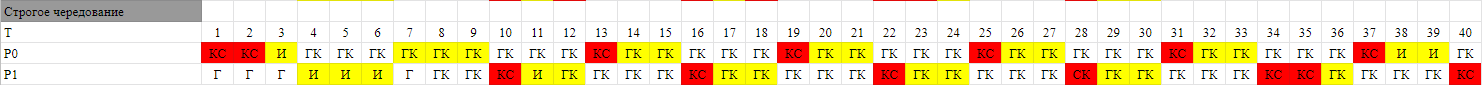


Рисунок 2 - Алгоритм «Строгое чередование» для P0 и P1

**3. Алгоритм взаимодействия трех процессов «Переменная-замок с 3 процессами» и «Строгое чередование с 3 процессами»**

Выполнить алгоритмы синхронизации процессов (*Р0, Р1*) «переменная – замок» и «строгое – чередование», использующих общие ресурсы, при наличии третьего процесса (*Р2*), не использующего ресурсы процессов *Р0*, *Р1*. Данные процессов (*Р0*, *Р1)* «приведенных в таблице 2.1, процесс *Р2* появляется каждый 6 квант времени, длительность процесса равна 3 квантам. Алгоритм планирования процессов *Round Robin (RR)*, величина кванта времени 3. Если процесс *Р2* выполниться не успел, новый его экземпляр в очередь не ставится. Процесс *Р2* не может прервать выполнение критической секции. Результаты оформить в виде таблиц иллюстрирующих работу процессов. Пример выполнения третьего задания указан на рисунке 3 и 4.

На рисунке 3 представлен результат выполнения 3 задания.

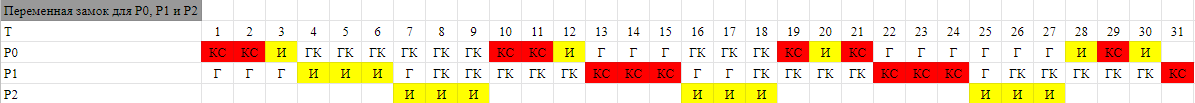
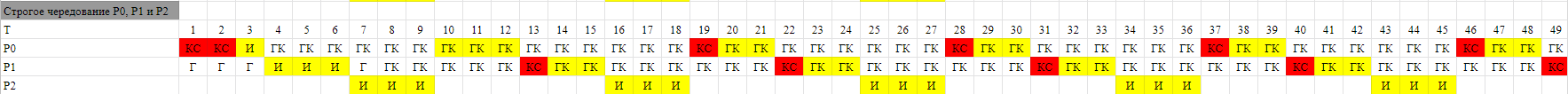


Рисунок 3 - Алгоритм «Переменная – замок» для P0, P1 и P2

На рисунке 4 представлен результат выполнения 3 задания.

****

****

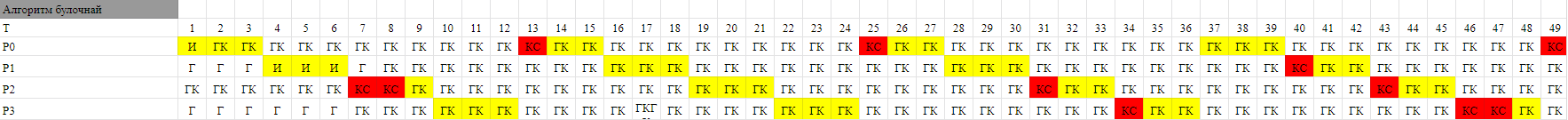
Рисунок 4 - Алгоритм «Строгое чередование» для P0, P1 и P2

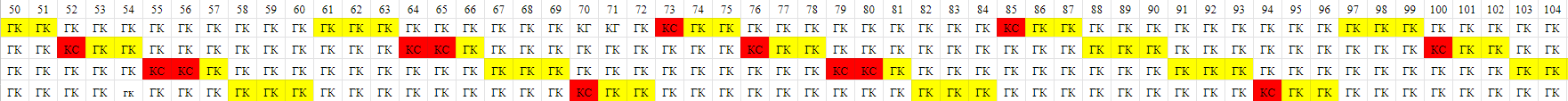
**4. Алгоритм взаимодействия нескольких процессов**

Выполнить алгоритм синхронизации четырех процессов (*Р0*, *Р1,* *Р2*, *Р3)* «алгоритм булочной», использующих общие ресурсы. Процессы выбираются из таблицы 2.1, согласно таблице 2.2. При каждой постановке в очередь критической секции, вычисляется номер, присваиваемый процессу. Алгоритм планирования процессов *Round Robin (RR*), величина кванта времени 3. Результаты оформить в виде таблицы, иллюстрирующей работу процессов, в таблице указывать номер. Пример выполнения четвёртого задания указан на рисунке 5.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PO | Время возникновения | 2 | 5 | 13 | 18 | 20 | 23 |
| Продолжительность | 1 | 1 | 1 | 1 | 1 | 1 |
| P1 | Время возникновения | 8 | 13 | 15 | 19 | 22 | 29 |
| Продолжительность | 1 | 1 | 2 | 1 | 1 | 2 |
| P2 | Время возникновения | 1 | 5 | 10 | 14 | 19 | 25 |
| Продолжительность | 2 | 1 | 1 | 2 | 2 | 1 |
| P3 | Время возникновения | 7 | 10 | 15 | 20 | 26 | 30 |
| Продолжительность | 1 | 2 | 1 | 1 | 2 | 1 |
|  |  |  |  |  |  |  |  |

На рисунке 5 представлен результат выполнения 4 задания.





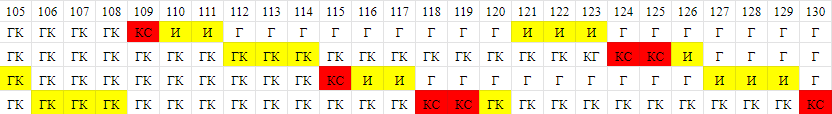


Рисунок 5 - «Алгоритм булочной» для P0, P1, P2 и P3

**Вывод:** в лабораторной работе были изучены и протестированы различные алгоритмы планирования процессов. Научились обрабатывать несколько процессов, и синхронизировать их. Выяснили что, чем проще алгоритм, тем меньше времени простоя процессора, однако они также не являются идеальными.