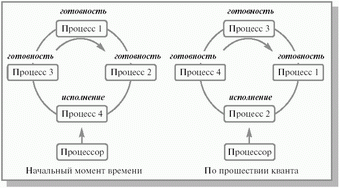
Практика № 5

**Алгоритм планирования - Round Robin (RR)**

Модификацией *алгоритма FCFS* является алгоритм, получивший название *Round Robin* (*Round Robin* – это вид детской карусели в США) или сокращенно ***RR***. По сути дела, это тот же самый алгоритм, только реализованный в режиме *вытесняющего планирования*. Можно представить себе все множество готовых процессов организованным циклически – процессы сидят на карусели. Карусель вращается так, что каждый процесс находится около процессора небольшой фиксированный *квант времени*, обычно 10 – 100 миллисекунд (см. рис. 1). Пока процесс находится рядом с процессором, он получает процессор в свое распоряжение и может исполняться.



**Рис. 1.** Процессы на карусели

Реализуется такой алгоритм так же, как и предыдущий, с помощью организации процессов, находящихся в состоянии готовность, в очередь FIFO. Планировщик выбирает для очередного исполнения процесс, расположенный в начале очереди, и устанавливает таймер для генерации прерывания по истечении определенного *кванта времени*. При выполнении процесса возможны два варианта.

* Время непрерывного использования процессора, необходимое процессу (остаток текущего *CPU burst* ), меньше или равно продолжительности *кванта времени*. Тогда процесс по своей воле освобождает процессор до истечения *кванта времени*, на исполнение поступает новый процесс из начала очереди, и таймер начинает отсчет *кванта* заново.
* Продолжительность остатка текущего *CPU burst* процесса больше, чем *квант времени*. Тогда по истечении этого *кванта* процесс прерывается таймером и помещается в конец очереди процессов, готовых к исполнению, а процессор выделяется для использования процессу, находящемуся в ее начале.

Рассмотрим предыдущий пример с порядком процессов p0, p1, p2 и величиной *кванта времени* равной 4. Выполнение этих процессов иллюстрируется таблицей 1. Обозначение "И" используется в ней для процесса, находящегося в состоянии исполнение, обозначение "Г" – для процессов в состоянии готовность, пустые ячейки соответствуют завершившимся процессам. Состояния процессов показаны на протяжении соответствующей единицы времени, т. е. колонка с номером 1 соответствует промежутку времени от 0 до 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 1. | | | | | | | | | | | | | | | | | | | |
| **Время** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| **p0** | И | И | И | И | Г | Г | Г | Г | Г | И | И | И | И | И | И | И | И | И |
| **p1** | Г | Г | Г | Г | И | И | И | И |  |  |  |  |  |  |  |  |  |  |
| **p2** | Г | Г | Г | Г | Г | Г | Г | Г | И |  |  |  |  |  |  |  |  |  |

Первым для исполнения выбирается процесс p0. Продолжительность его *CPU burst* больше, чем величина *кванта времени*, и поэтому процесс исполняется до истечения *кванта*, т. е. в течение 4 единиц времени. После этого он помещается в конец очереди готовых к исполнению процессов, которая принимает вид p1, p2, p0. Следующим начинает выполняться процесс p1. Время его исполнения совпадает с величиной выделенного *кванта*, поэтому процесс работает до своего завершения. Теперь очередь процессов в состоянии готовность состоит из двух процессов, p2 и p0. Процессор выделяется процессу p2. Он завершается до истечения отпущенного ему процессорного времени, и очередные *кванты* отмеряются процессу p0 – единственному не закончившему к этому моменту свою работу. Время ожидания для процесса p0 (количество символов "Г" в соответствующей строке) составляет 5 единиц времени, для процесса p1 – 4 единицы времени, для процесса p2 – 8 единиц времени. Таким образом, среднее время ожидания для этого алгоритма получается равным (5 + 4 + 8)/3 = 5,6(6) единицы времени. Полное время выполнения для процесса p0 (количество непустых столбцов в соответствующей строке) составляет 18 единиц времени, для процесса p1 – 8 единиц, для процесса p2 – 9 единиц. Среднее полное время выполнения оказывается равным (18 + 8 + 9)/3 = 11,6(6) единицы времени.

Легко увидеть, что среднее время ожидания и среднее полное время выполнения для обратного порядка процессов не отличаются от соответствующих времен для *алгоритма FCFS* и составляют 2 и 8 единиц времени соответственно.

На производительность *алгоритма RR* сильно влияет величина *кванта времени*. Рассмотрим тот же самый пример с порядком процессов p0, p1, p2 для величины *кванта времени*, равной 1 (см. табл. 2). Время ожидания для процесса p0 составит 5 единиц времени, для процесса p1 – тоже 5 единиц, для процесса p2 – 2 единицы. В этом случае среднее время ожидания получается равным (5 + 5 + 2)/3 = 4 единицам времени. Среднее полное время исполнения составит (18 + 9 + 3)/3 = 10 единиц времени.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 2 | | | | | | | | | | | | | | | | | | | |
| **Время** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| **p0** | И | Г | Г | И | Г | И | Г | И | Г | И | И | И | И | И | И | И | И | И |
| **p1** | Г | И | Г | Г | И | Г | И | Г | И |  |  |  |  |  |  |  |  |  |
| **p2** | Г | Г | И |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

При очень больших величинах *кванта времени*, когда каждый процесс успевает завершить свой *CPU burst* до возникновения прерывания по времени, *алгоритм RR* вырождается в *алгоритм FCFS*. При очень малых величинах создается иллюзия того, что каждый из n процессов работает на собственном виртуальном процессоре с производительностью ~ 1/n от производительности реального процессора. Правда, это справедливо лишь при теоретическом анализе при условии пренебрежения временами переключения *контекста процессов*. В реальных условиях при слишком малой величине *кванта времени* и, соответственно, слишком частом переключении контекста накладные расходы на переключение резко снижают производительность системы

**Задания**.

Максимальное количество баллов за данную практику - 5. Решение каждого из заданий с учетом положительных ответов на вопросы при защите – 1 балл.

**Задание №1**

Алгоритм RR. Построить таблицу выполнения процессов. Выполнить расчет среднего времени ожидания и полного времени выполнения при исполнении на квант времени = 3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Процесс | Р0 | Р1 | Р2 | P3 | P4 | P5 |
| Продолжительность очередного CPU burst | 1 | 3 | 12 | 10 | 5 | 2 |

**Задание №2**

Алгоритм RR. Построить таблицу выполнения процессов. Выполнить расчет среднего времени ожидания и полного времени выполнения при исполнении на квант времени = 2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Процесс | Р0 | Р1 | Р2 | P3 | P4 | P5 |
| Продолжительность очередного CPU burst | 8 | 12 | 6 | 10 | 3 | 4 |

**Задание №3**

Алгоритм RR. Построить таблицу выполнения процессов. Выполнить расчет среднего времени ожидания и полного времени выполнения при исполнении на квант времени = 2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Процесс | Р0 | Р1 | Р2 | P3 | P4 | P5 |
| Продолжительность очередного CPU burst | 12 | 7 | 4 | 9 | 6 | 2 |

**Задание №4**

Алгоритм RR. Построить таблицу выполнения процессов. Выполнить расчет среднего времени ожидания и полного времени выполнения при исполнении на квант времени = 4.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Процесс | Р0 | Р1 | Р2 | P3 | P4 | P5 |
| Продолжительность очередного CPU burst | 6 | 9 | 3 | 12 | 10 | 7 |

**Задание №5**

Выполнить программную реализацию алгоритма планирования процессов Round Robin. Программное средство должно выполнять все расчеты, которые проводились в методическом пособии.