**Алгоритмы планирования**

Существует достаточно большой набор разнообразных алгоритмов *планирования*, которые предназначены для достижения различных целей и эффективны для разных классов задач. Многие из них могут использоваться на нескольких *уровнях планирования*. В этом разделе мы рассмотрим некоторые наиболее употребительные алгоритмы применительно к процессу кратковременного *планирования*.

**First-Come, First-Served (FCFS)**

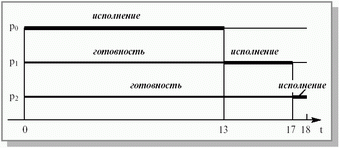
Простейшим алгоритмом *планирования* является алгоритм, который принято обозначать аббревиатурой ***FCFS*** по первым буквам его английского названия – First-Come, First-Served (первым пришел, первым обслужен). Представим себе, что процессы, находящиеся в состоянии готовность, выстроены в очередь. Когда процесс переходит в состояние готовность, он, а точнее, ссылка на его *PCB* помещается в конец этой очереди. Выбор нового процесса для исполнения осуществляется из начала очереди с удалением оттуда ссылки на его *PCB*. Очередь подобного типа имеет в программировании специальное наименование – FIFO1, сокращение от First In, First Out (первым вошел, первым вышел).

Такой алгоритм выбора процесса осуществляет *невытесняющее планирование*. Процесс, получивший в свое распоряжение процессор, занимает его до истечения текущего *CPU burst* . После этого для выполнения выбирается новый процесс из начала очереди.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица 1. | | | | |
| **Процесс** | p0 | p1 | p2 |
| **Продолжительность очередного** ***CPU burst*** | 13 | 4 | 1 |

Преимуществом *алгоритма FCFS* является легкость его реализации, но в то же время он имеет и много недостатков. Рассмотрим следующий пример. Пусть в состоянии готовность находятся три процесса p0, p1 и p2, для которых известны времена их очередных *CPU burst* . Эти времена приведены в [таблице 1.](https://intuit.ru/studies/professional_retraining/941/courses/31/lecture/972?page=3#table.3.1) в некоторых условных единицах. Для простоты будем полагать, что вся деятельность процессов ограничивается использованием только одного промежутка *CPU burst* , что процессы не совершают операций ввода-вывода и что время переключения контекста так мало, что им можно пренебречь.

Если процессы расположены в очереди процессов, готовых к исполнению, в порядке p0, p1, p2, то картина их выполнения выглядит так, как показано на рисунке 1. Первым для выполнения выбирается процесс p0, который получает процессор на все время своего *CPU burst* , т. е. на 13 единиц времени. После его окончания в состояние исполнение переводится процесс p1, он занимает процессор на 4 единицы времени. И, наконец, возможность работать получает процесс p2. Время ожидания для процесса p0 составляет 0 единиц времени, для процесса p1 – 13 единиц, для процесса p2 – 13 + 4 = 17 единиц. Таким образом, среднее время ожидания в этом случае – (0 + 13 + 17)/3 = 10 единиц времени. Полное время выполнения для процесса p0 составляет 13 единиц времени, для процесса p1 – 13 + 4 = 17 единиц, для процесса p2 – 13 + 4 + 1 = 18 единиц. Среднее полное время выполнения оказывается равным (13 + 17 + 18)/3 = 16 единицам времени.



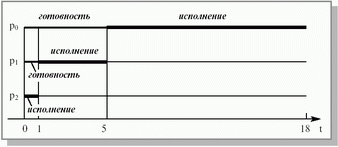
**Рис. 1.** Выполнение процессов при порядке p0, p1,p2

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

***Г – готовность;***

***И – исполнение.***

Если те же самые процессы расположены в порядке p2, p1, p0, то картина их выполнения будет соответствовать рисунку 2. Время ожидания для процесса p0 равняется 5 единицам времени, для процесса p1 – 1 единице, для процесса p2 – 0 единиц. Среднее время ожидания составит (5 + 1 + 0)/3 = 2 единицы времени. Это в 5 (!) раз меньше, чем в предыдущем случае. Полное время выполнения для процесса p0 получается равным 18 единицам времени, для процесса p1 – 5 единицам, для процесса p2 – 1 единице. Среднее полное время выполнения составляет (18 + 5 + 1)/3 = 8 единиц времени, что почти в 2 раза меньше, чем при первой расстановке процессов.



**Рис. 2.** Выполнение процессов при порядке p2, p1, p0

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

Как мы видим, среднее время ожидания и среднее полное время выполнения для этого алгоритма существенно зависят от порядка расположения процессов в очереди. Если у нас есть процесс с длительным *CPU burst* , то короткие процессы, перешедшие в состояние готовность после длительного процесса, будут очень долго ждать начала выполнения. Поэтому *алгоритм FCFS* практически неприменим для систем разделения времени – слишком большим получается среднее время отклика в *интерактивных процессах*.

**Задание на практикум 4 *(каждое задание оценивается в 1 балл с учетом корректного выполнения и ответов на вопросы по исполнению. Максимальное количество баллов за практикум - 3).***

1. Алгоритм FCFS. Построить таблицу выполнения процессов. Выполнить расчет среднего времени ожидания и полного времени выполнения при исполнении:

p0, p1, p2, p3, p4, p5;

p5, p4, p3, p2, p1, p0;

предложить оптимальный порядок исполнения.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Процесс | Р0 | Р1 | Р2 | P3 | P4 | P5 |
| Продолжительность очередного CPU burst | 3 | 9 | 3 | 5 | 3 | 1 |

1. Алгоритм FCFS. Построить таблицу выполнения процессов. Выполнить расчет среднего времени ожидания и полного времени выполнения при исполнении:

p0, p1, p2, p3, p4, p5;

p5, p4, p3, p2, p1, p0;

предложить оптимальный порядок исполнения.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Процесс | p0 | p1 | p2 | P3 | P4 | P5 |
| Продолжительность очередного CPU burst | 7 | 1 | 4 | 9 | 4 | 5 |

1. Выполнить программную реализацию алгоритма FCFS