**Алгоритмы планирования**

**Гарантированное планирование**

При интерактивной работе N пользователей в вычислительной системе можно применить алгоритм планирования, который гарантирует, что каждый из пользователей будет иметь в своем распоряжении ~1/N часть процессорного времени.

Пронумеруем всех пользователей от 1 до N.

Для каждого пользователя с номером i введем две величины: Ti – время нахождения пользователя в системе или, другими словами, длительность сеанса его общения с машиной и \tau_i – суммарное процессорное время уже выделенное всем его процессам в течение сеанса.

\tau_i >> T_i/N\tau_i << T_i/NСправедливым для пользователя было бы получение Ti/N процессорного времени. Если то i -й пользователь несправедливо обделен процессорным временем.

\tau_i N/T_iЕсли же то система явно благоволит к пользователю с номером i. Вычислим для процессов каждого пользователя значение коэффициента справедливости и будем предоставлять очередной квант времени готовому процессу с наименьшей величиной этого отношения.

Предложенный алгоритм называют алгоритмом гарантированного планирования.

К недостаткам этого алгоритма можно отнести невозможность предугадать поведение пользователей.

**Приоритетное планирование**

При приоритетном планировании каждому процессу присваивается число – приоритет, в соответствии с которым ему выделяется процессор.

Процессы с одинаковыми приоритетами планируются в порядке FCFS.

Алгоритм опирается как на внутренние, так и на внешние параметры.

Для алгоритма SJF в качестве такого приоритета выступает оценка продолжительности следующего CPU burst. Чем меньше значение этой оценки, тем более высокий приоритет имеет процесс. Для алгоритма гарантированного планирования приоритетом служит вычисленный коэффициент справедливости. Чем он меньше, тем больше у процесса приоритет.

Внутренние параметры:

* Ограничение по времени использования процессора
* Требование к размеру памяти
* Число открытых файлов в системе
* Число устройств ввода-вывода и т.д.

Внешние параметры:

* Важность процесса для пользователя
* Достижение каких-либо целей в системе и т.д.

Планирование с использованием процессора может быть как вытесняющим, так и не вытесняющим.

При вытесняющем режиме процесс с более высоким приоритетом, появившийся в очереди готовых процессов, вытесняет исполняющийся процесс с более низким приоритетом. В случае не вытесняющего планирования он просто становится в начало очереди готовых процессов.

Приоритетные алгоритмы делятся на 2 группы:

* Статические
* Динамические

В статических приоритет процесса с течением времени не меняется.

В динамических приоритет со временем изменяется.

Механизмы статической приоритетности легко реализуются и они сопряжены с небольшими издержками, однако они не реагируют на изменение ситуации в системе.

Динамическое планирование , его алгоритмы, более сложное и гибкие.

Изменение приоритета происходит согласованно с совершением каких-либо других операций над процессами.

Проблема – низкоприоритетные процессы могут не запускаться в течение долгого времени.

Пути решения этой проблемы:

1. С течением времени повышать приоритет у низкоприоритетных процессов, находящихся в состоянии готовности.

**Многоуровневые очереди**

Для систем, в которых процессы могут быть легко рассортированы по разным группам, был разработан другой класс алгоритмов планирования.

Для каждой группы процессов создается своя очередь процессов, находящихся в состоянии готовность.

При этом очередям приписывают фиксированные приоритеты.

Внутри этих очередей для планирования могут применяться самые разные алгоритмы.

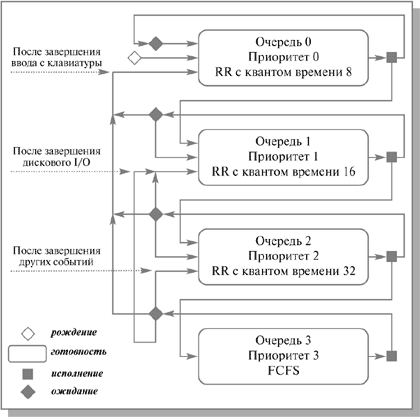
Так, например, для больших счетных процессов, не требующих взаимодействия с пользователем (фоновых процессов), может использоваться алгоритм FCFS, а для интерактивных процессов – алгоритм RR.

Подобный подход получивший название многоуровневых очередей, повышает гибкость планирования: для процессов с различными характеристиками применяется наиболее подходящим им алгоритм.

**Многоуровневые очереди с обратной связью**

Дальнейшим развитием алгоритма многоуровневых очередей является добавление к нему механизма обратной связи. Здесь процесс не постоянно приписан к определенной очереди, а может мигрировать из одной очереди в другую в зависимости от своего поведения.

В качестве примера рассмотрим следующую схему (схема миграции процессов в многоуровневых очередях планирования с обратной связью) :



Многоуровневые очереди с обратной связью – наиболее общий подход из всех. Они наиболее трудны в своей реализации, но и наиболее гибки. Существует большое количество разновидностей такого способа планирования.

Для полного описания и полного упрощения необходимо указать:

* Количество очередей
* Описать алгоритм планирования, действующий между очередями
* Указать алгоритм планирования, действующий в каждой очереди
* Описать правило появления новых процессов в одну из очередей
* Описать правило перевода процесса из одной очереди в другую

Изменяя какой-либо из этих пунктов можно существенно изменять поведение всей ВС.