3.14. Планирование выполнения процессов.

Алгоритмы (карусельные, учитывающие приоритеты, учитывающие взаимодействие с ресурсами).

Real-time планирование.

Планирование на нескольких вычислительных узлах, хищение и добровольная миграция процессов.

Синдром обмена приоритетами, методы его избежания.

Уровни планирования

Планирование заданий используется в качестве **долгосрочного** планирования процессов. Оно отвечает за порождение новых процессов в системе. Долгосрочное планирование осуществляется достаточно редко, между появлением новых процессов могут проходить минуты и даже десятки минут.
Планирование использования процессора применяется в качестве **краткосрочного** планирования процессов. Краткосрочное планирование осуществляется, как правило, не реже одного раза в 100 миллисекунд.

В некоторых вычислительных системах бывает выгодно для повышения производительности временно удалить какой-либо частично выполнившийся процесс из оперативной памяти на диск, а позже вернуть его обратно для дальнейшего выполнения. Такая процедура – *swapping, "перекачка", свопинг*. Когда и какой из процессов нужно перекачать на диск и вернуть обратно, решается дополнительным промежуточным уровнем планирования процессов – **среднесрочным**.

Критерии планирования и требования к алгоритмам

Цели, которые мы хотим достичь, используя планирование:

- Справедливость – гарантировать каждому заданию или процессу определенную часть времени использования процессор, стараясь не допустить ситуации, когда один процесс все «кушает», другой «нервно курит в стороне».
- Эффективность – постараться занять процессор на все 100%
- Сокращение полного времени выполнения (turnaround time) – обеспечить минимальное время между стартом процесса и его завершением.
- Сокращение времени ожидания (waiting time) – сократить время, которое проводят процессы в состоянии готовность.
- Сокращение времени отклика (response time) – минимизировать время, которое требуется процессу в интерактивных системах для ответа на запрос пользователя.

Желательно, чтобы алгоритмы обладали следующими свойствами:

- Были предсказуемыми. Одно и то же задание должно выполняться приблизительно за одно и то же время.
- Минимальные накладные расходы.
- Равномерно загружали ресурсы вычислительной системы, отдавая предпочтение тем процессам, которые будут занимать малоиспользуемые ресурсы.
- Обладали масштабируемостью.

Вытесняющее и невытесняющее планирование

Невытесняющее планирование используется, например, в MS Windows 3.1 и ОС Apple Macintosh. При таком режиме планирования процесс занимает столько процессорного времени, сколько ему необходимо. При этом переключение процессов возникает только при желании самого исполняющегося процесса передать управление (для ожидания завершения операции ввода-вывода или по окончании работы).

Этот метод планирования относительно просто реализуем и достаточно эффективен, так как позволяет выделить большую часть процессорного времени для работы самих процессов и до минимума сократить затраты на переключение контекста.

Однако при невытесняющем планировании возникает проблема возможности полного захвата процессора одним процессом.

Вытесняющее планирование обычно используется в системах разделения времени. В этом режиме планирования процесс может быть приостановлен в любой момент исполнения.

Временные прерывания помогают гарантировать приемлемое время отклика процессов для пользователей, работающих в диалоговом режиме, и предотвращают "зависание" компьютерной системы.

Алгоритмы краткосрочного планирования

First-Come, First-Served (FCFS)

First-Come, First-Served (первым пришел, первым обслужен) на очереди FIFO

Представим себе, что процессы, находящиеся в состоянии готовность, выстроены в очередь. Когда процесс переходит в состояние готовность, он помещается в конец этой очереди. Выбор нового процесса для исполнения осуществляется из начала очереди с удалением оттуда.

Такой алгоритм выбора процесса осуществляет **невытесняющее** планирование.

Round Robin (RR)

Модификация алгоритма FCFS. (Round Robin – это вид детской карусели в США)

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. Процесс в начале исполняется некоторое время, потом, если он не завершился, помещается в конец очереди.

Shortest-Job-First (SJF)

Упорядочим все процессы по необходимому процессорному времени, до следующего перерыва (работа с ресурсами, ожидание других, …). И будем брать всегда с минимальным временем.

Квантование времени при этом не применяется. Описанный алгоритм получил название "кратчайшая работа первой" или Shortest Job First (SJF).

SJF-алгоритм краткосрочного планирования может быть как **вытесняющим**, так и **невытесняющим**. При невытесняющем SJF-планировании процессор предоставляется избранному процессу на все необходимое ему время. При вытесняющем SJF-планировании учитывается появление новых процессов в очереди готовых к исполнению.

Гарантированное планирование

N пользователей. Хотим: каждый из пользователей имеет в своем распоряжении ~1/N часть процессорного времени.

T_i – время нахождения пользователя в системе или, другими словами, длительность сеанса его общения с машиной и τ_i – суммарное процессорное время уже выделенное всем его процессам в течение сеанса.

Вычислим для процессов каждого пользователя значение коэффициента справедливости

$\tau_i N/T_i$

и будем предоставлять очередной квант времени готовому процессу с наименьшей величиной этого отношения.

Приоритетное планирование

При приоритетном планировании каждому процессу присваивается определенное числовое значение – приоритет, в соответствии с которым ему выделяется процессор. Процессы с одинаковыми приоритетами планируются в порядке FCFS.

Планирование с использованием приоритетов может быть как **вытесняющим**, так и **невытесняющим**. При вытесняющем планировании процесс с более высоким приоритетом, появившийся в очереди готовых процессов, вытесняет исполняющийся процесс с более низким приоритетом. В случае невытесняющего планирования он просто становится в начало очереди готовых процессов.

Главная проблема приоритетного планирования заключается в том, что при ненадлежащем выборе механизма назначения и изменения приоритетов низкоприоритетные процессы могут не запускаться неопределенно долгое время.

Решение этой проблемы может быть достигнуто с помощью увеличения со временем значения приоритета процесса, находящегося в состоянии готовность.

Многоуровневые очереди (Multilevel Queue)

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. Этим очередям приписываются фиксированные приоритеты.

Например, приоритет очереди системных процессов устанавливается выше, чем приоритет очередей пользовательских процессов. Это значит, что ни один пользовательский процесс не будет выбран для исполнения, пока есть хоть один готовый системный процесс.

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 подходящий им алгоритм.

Многоуровневые очереди с обратной связью (Multilevel Feedback Queue)

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.