Планирование

Планирование

Когда компьютер работает в многозадачном режиме, на нем зачастую запускается сразу несколько процессов или потоков, претендующих на использование центрального процессора. Такая ситуация складывается в том случае, если в состоянии готовности одновременно находятся два или более процесса или потока.

Планирование

Если доступен только один центральный процессор, необходимо выбрать, какой из этих процессов будет выполняться следующим.

Та часть операционной системы, на которую возложен этот выбор, называется **планировщиком**, а алгоритм, который ею используется, называется **алгоритмом планирования**.

Планирование

Если доступен только один центральный процессор, необходимо выбрать, какой из этих процессов будет выполняться следующим.

Та часть операционной системы, на которую возложен этот выбор, называется **планировщиком**, а алгоритм, который ею используется, называется **алгоритмом планирования**.

Как было раньше

Если вернуться к прежним временам пакетных систем, где ввод данных осуществлялся в форме образов перфокарт, перенесенных на магнитную ленту, то алгоритм планирования был довольно прост: требовалось всего лишь запустить следующее имеющееся на ленте задание.

Как потом

С появлением многозадачных систем алгоритм планирования усложнился, поскольку в этом случае обычно фигурировали сразу несколько пользователей, ожидавших обслуживания.

С появлением ПК

С появлением персональных компьютеров ситуация изменилась в двух направлениях. Во-первых, основная часть времени отводилась лишь одному активному процессу.

С появлением ПК

Во-вторых, с годами компьютеры стали работать настолько быстрее, что центральный процессор практически перестал быть дефицитным ресурсом. Большинство программ для персонального компьютера ограничены скоростью предоставления пользователем входящей информации (путем набора текста или щелчками мыши), а не скоростью, с которой центральный процессор способен ее обработать.

Планирование служб

Когда же дело касается сетевых служб, ситуация существенно изменяется. Здесь в конкурентную борьбу за процессорное время вступает уже несколько процессов, поэтому планирование снова приобретает значение. Например, когда центральному процессору нужно выбирать между запущенным процессом, собирающим ежедневную статистику, и одним из процессов, обслуживающих запросы пользователя, если приоритет будет сразу же отдан последнему из процессов, пользователь будет

Переключение между процессами

Планировщик, кроме выбора «правильного» процесса, должен также заботиться об эффективной загрузке CPU, поскольку переключение процессов является весьма дорогостоящим занятием.

Сначала должно произойти переключение из пользовательского режима в режим ядра. Затем должно быть сохранено состояние текущего процесса, включая сохранение его регистров в таблице процессов для их последующей повторной загрузки.

Переключение между процессами

После этого запускается алгоритм планирования для выбора следующего процесса.

Затем в соответствии с картой памяти нового процесса должен быть перезагружен блок управления памятью. И наконец, новый процесс должен быть запущен.

Переключение между процессами

Вдобавок ко всему перечисленному переключение процессов обесценивает весь кэш памяти, заставляя его дважды динамически перезагружаться из оперативной памяти (после входа в ядро и после выхода из него). В итоге слишком частое переключение может поглотить существенную долю процессорного времени, что наводит на мысль о том, что этого нужно избегать.

Поведение процесса

Практически у всех процессов чередуются пики вычислительной активности с запросами (дискового) ввода-вывода. Обычно центральный процессор некоторое время работает без остановок, затем происходит системный вызов для чтения данных из файла или для их записи в файл.

Поведение процесса

Когда системный вызов завершается, центральный процессор возобновляет вычисления до тех пор, пока ему не понадобятся дополнительные данные или не потребуется записать дополнительные данные на диск и т. д.

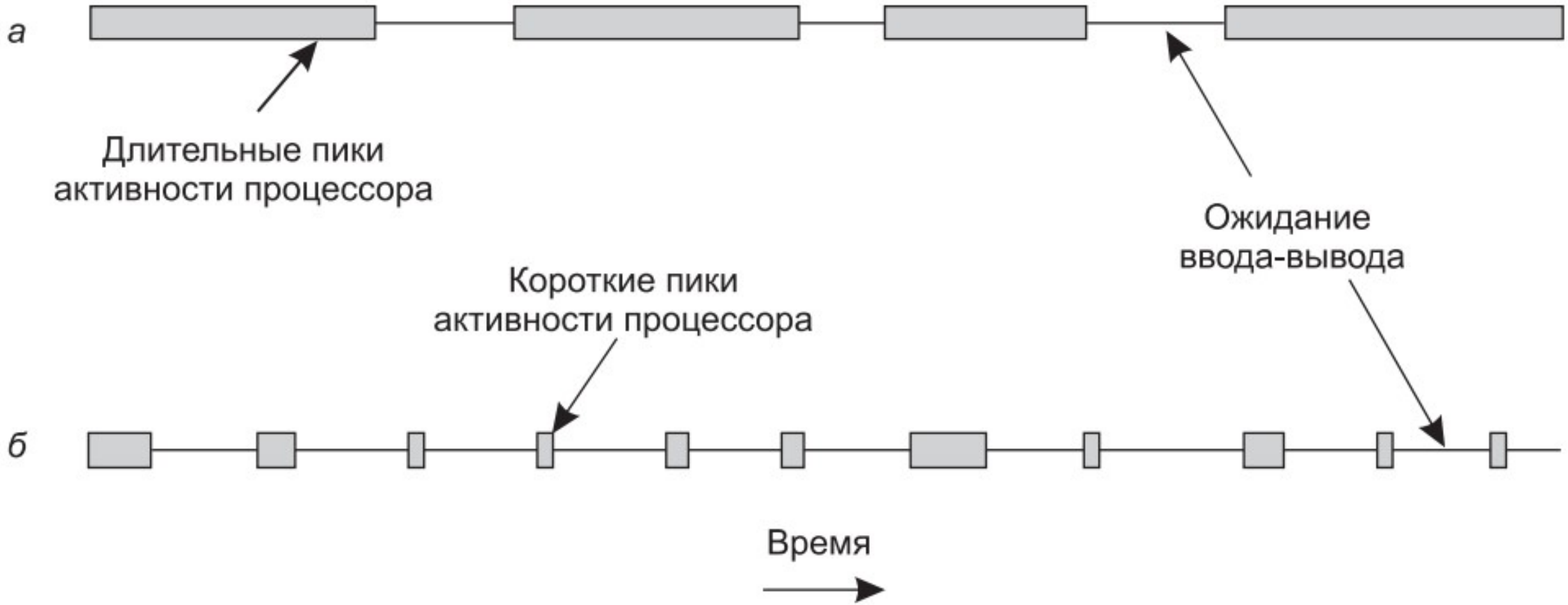


Рис. 2.19. Пики активного использования центрального процессора чередуются с периодами ожидания завершения операций ввода-вывода: процесс, ограниченный скоростью вычислений (*а*); процесс, ограниченный скоростью работы устройств ввода-вывода (*б*)

Поведение процесса

Некоторые процессы, как тот, что показан на рис. 2.1, *а*, проводят основную часть своего времени за вычислениями, а другие, как тот, что показан на рис. 2.1, *б*, проводят основную часть своего времени в ожидании завершения операций ввода-вывода.

Первые процессы называются процессами, **ограниченными скоростью вычислений**, а вторые — процессами, **ограниченными скоростью работы устройств ввода-вывода**.

Поведение процесса

Чем быстрее становятся центральные процессоры, тем больше процессы ограничиваются скоростью работы устройств ввода-вывода

Это связано с более быстрым совершенствованием центральных процессоров по сравнению с совершенствованием дисковых устройств.

Когда планировать

Во-первых, при создании нового процесса необходимо принять решение, какой из процессов выполнять: родительский или дочерний.

Во-вторых, планировщик должен принять решение, когда процесс завершает работу. Процесс больше не может выполняться поэтому нужно выбрать какой-нибудь процесс из числа готовых к выполнению.

В-третьих, когда процесс блокируется в ожидании завершения операции ввода_x0002_вывода, на семафоре или по какой-нибудь другой причине, для выполнения должен быть выбран какой-нибудь другой процесс

Когда планировать

В-четвертых, планировщик должен принять решение при возникновении прерывания ввода-вывода. Если прерывание пришло от устройства ввода-вывода, завершившего свою работу, то какой-то процесс, который был заблокирован в ожидании завершения операции ввода-вывода, теперь может быть готов к возобновлению работы.

Категории алгоритмов планирования

В различных условиях окружающей среды требуются разные алгоритмы планирования. Это обусловлено тем, что различные сферы приложений (и разные типы операционных систем) предназначены для решения разных задач

Категории алгоритмов планирования

Существует три основных среды:

1) пакетную;

2) интерактивную;

3) реального времени.

Категории алгоритмов планирования

Существует три основных среды:

1) пакетную;

2) интерактивную;

3) реального времени.

Пакетные системы

В пакетных системах не бывает пользователей, терпеливо ожидающих за своими терминалами быстрого ответа на свой короткий запрос. Поэтому для них зачастую приемлемы неприоритетные алгоритмы или приоритетные алгоритмы с длительными периодами для каждого процесса. Такой подход сокращает количество переключений между процессами, повышая при этом производительность работы системы.

Интерактивные системы

В среде с пользователями, работающими в интерактивном режиме, приобретает важность приоритетность, сдерживающая отдельный процесс от захвата центрального процессора.

Системы реального времени

В системах, ограниченных условиями реального времени, приоритетность иногда не требуется, поскольку процессы знают, что они могут запускаться только на непродолжительные периоды времени, и зачастую выполняют свою работу довольно быстро, а затем блокируются.

Вопросы

1. Зачем нужен планировщик заданий?
2. Что такое пакетные системы?
3. Что такое интерактивные системы?
4. Что такое неприоритетные алгоритмы?
5. Что такое квант времени?

Задачи алгоритма планирования

Пакетные системы:

- Производительность — выполнение максимального количества заданий в час.

- Оборотное время — минимизация времени между представлением задачи и ее завершением.

- Использование центрального процессора — поддержка постоянной загруженности процессора.

Задачи алгоритма планирования

Интерактивные системы:

- Время отклика — быстрый ответ на запросы.

- Пропорциональность — оправдание пользовательских надежд.

Задачи алгоритма планирования

Системы реального времени:

- Соблюдение предельных сроков — предотвращение потери данных.

- Предсказуемость — предотвращение ухудшения качества в мультимедийных системах.

Планирование в пакетных системах

Первым пришел — первым обслужен

При использовании этого алгоритма центральный процессор выделяется процессам в порядке поступления их запросов. По сути, используется одна очередь процессов, находящихся в состоянии готовности

Первым пришел — первым обслужен

Когда рано утром в систему извне попадает первое задание, оно тут же запускается на выполнение и получает возможность выполняться как угодно долго. Оно не прерывается по причине слишком продолжительного выполнения. По мере поступления других заданий они помещаются в конец очереди. При блокировке выполняемого процесса следующим запускается первый процесс, стоящий в очереди. Когда заблокированный процесс переходит в состояние готовности, он, подобно только что поступившему заданию, помещается в конец очереди.

Сначала самое короткое задание

Неприоритетный алгоритм для пакетных систем, в котором предполагается, что сроки выполнения заданий известны заранее.

Сначала самое короткое задание

Здесь представлены четыре задания: A, B, C и D со сроками выполнения 8, 4, 4 и 4 минуты соответственно. Если их запустить в этом порядке, оборотное время для задания A составит 8 мин, для B — 12 мин, для C — 16 мин и для D — 20 мин, при среднем времени 14 мин.

Сначала самое короткое задание

Рис. 2.20. Пример планирования, когда первым выполняется самое короткое задание: запуск четырех заданий в исходном порядке (*а*); запуск этих заданий в порядке, когда самое короткое из них выполняется первым (*б*)

Сначала самое короткое задание

Первое задание будет выполнено за время a, второе — за время a + b, и т. д. Среднее оборотное время составит (4a + 3b + 2c + d)/4. Очевидно, что время a оказывает наибольшее влияние на средний показатель по сравнению со всеми остальными временными показателями, поэтому это должно быть самое короткое задание, затем по нарастающей должны идти b, c и, наконец, d

Приоритет наименьшему времени выполнения

Приоритетной версией алгоритма выполнения первым самого короткого задания является алгоритм первоочередного выполнения задания с наименьшим оставшимся временем выполнения.

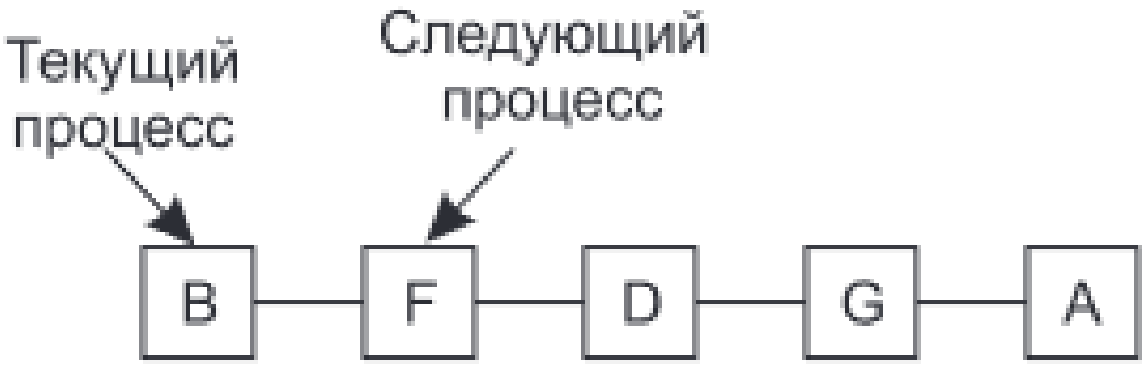
Планирование в интерактивных системах

Циклическое планирование

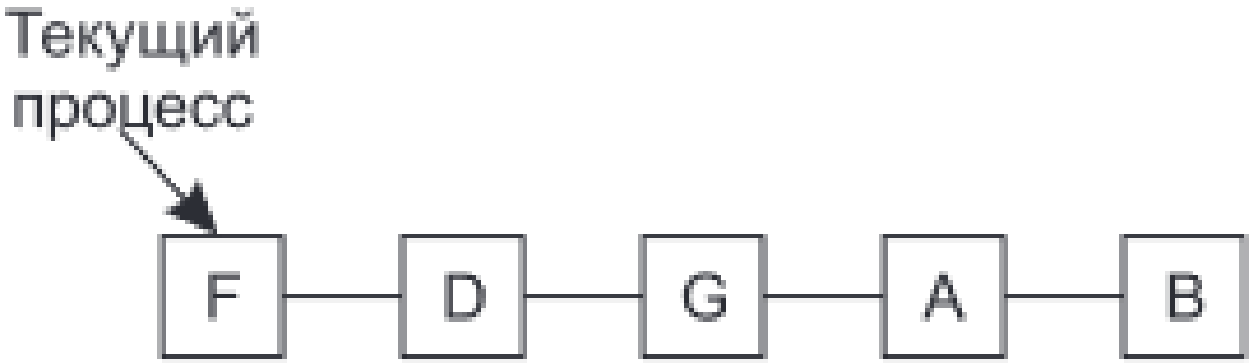
Каждому процессу назначается определенный интервал времени, называемый его квантом, в течение которого ему предоставляется возможность выполнения. Если процесс к завершению кванта времени все еще выполняется, то ресурс центрального процессора у него отбирается и передается другому процессу

Циклическое планирование

Если процесс переходит в заблокированное состояние или завершает свою работу до истечения кванта времени, то переключение центрального процессора на другой процесс происходит именно в этот момент.



Циклическое планирование: список процессов, находящихся в состоянии готовности



тот же список после того, как процесс B исчерпал свой квант времени

Циклическое планирование

Предположим, что переключение процесса занимает 1 мс. Также предположим, что значение кванта времени установлено на 4 мс

При таких параметрах настройки после 4 мс полезной работы центральному процессору придется затратить (то есть потерять) 1 мс на переключение процесса

Циклическое планирование

Таким образом, 20% процессорного времени будет выброшено на административные издержки, а это, вне всякого сомнения, слишком много.

Циклическое планирование

Чтобы повысить эффективность использования центрального процессора попробуес установить значение кванта 100 мс. Теперь теряется всего 1% времени. Но посмотрим, что получится на серверной системе,если к ней поступит за очень короткий отрезок времени 50 запросов, имеющих широкий разброс в степени востребованности центрального процессора.

Циклическое планирование

В список готовых к запуску процессов будет помещено 50 процессов. Первый из них будет запущен немедленно, второй не сможет запуститься, пока не истекут 100 мс, и т. д. Если предположить, что все процессы полностью воспользуются своими квантами времени, то самый невезучий последний процесс может пребывать в ожидании в течение 5 с, прежде чем получит шанс на запуск.

Циклическое планирование

Многим пользователям работа системы при пятисекундном ожидании ответа на короткую команду покажется слишком медленной.

Если квант времени будет короче, качество их обслуживания улучшится.

Зачастую разумным компромиссом считается квант времени в 20–50 мс.

Приоритетное планирование

:Каждому процессу присваивается значение приоритетности, и запускается тот процесс, который находится в состоянии готовности и имеет наивысший приоритет.

Приоритетное планирование

Даже если у ПК один владелец, на нем могут выполняться несколько процессов с разной степенью важности. Например, **фоновому процессу**, отправляющему электронную почту, должен быть назначен более низкий приоритет, чем процессу, **воспроизводящему на экране видеофильм в реальном времени**

Приоритетное планирование

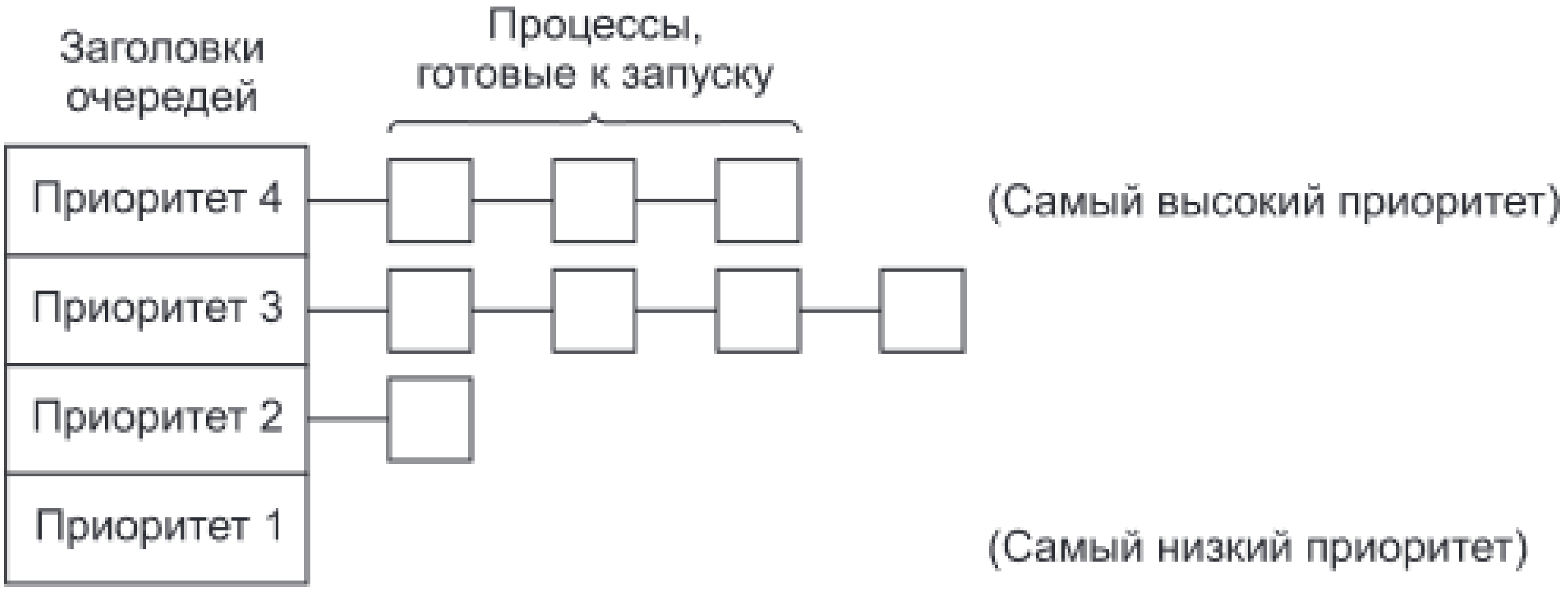
Чтобы предотвратить бесконечное выполнение высокоприоритетных процессов, планировщик должен понижать уровень приоритета текущего выполняемого процесса.

Приоритетное планирование

Другой вариант: каждому процессу может быть выделен максимальный квант допустимого времени выполнения. Когда квант времени будет исчерпан, шанс запуска будет предоставлен другому процессу, имеющему наивысший приоритет

Использование нескольких очередей

Зачастую бывает удобно группировать процессы по классам приоритетности и использовать приоритетное планирование применительно к этим классам, а внутри каждого класса использовать циклическое планирование.


Заголовки очередей
Процессы, готовые к запуску
Приоритет 4
Приоритет 3
Приоритет 2
Приоритет 1
(Самый высокий приоритет)
(Самый низкий приоритет)

Гарантийное планирование

Если в процессе работы в системе зарегистрировано n пользователей, то вы получите 1/n от мощности центрального процессора.

Лотерейное планирование

Основная идея состоит в раздаче процессам лотерейных билетов на доступ к различным системным ресурсам, в том числе и к процессорному времени. Когда планировщику нужно принимать решение, в случайном порядке выбирается лотерейный билет, и ресурс отдается процессу, обладающему этим билетом.

Лотерейное планирование

Применительно к планированию процессорного времени система может проводить лотерейный розыгрыш 50 раз в секунду, и каждый победитель будет получать в качестве приза 20 мс процессорного времени.

Лотерейное планирование

«Все процессы равны, но некоторые из них равнее остальных»

Лотерейное планирование

Более важным процессам, чтобы повысить их шансы на выигрыш, могут выдаваться дополнительные билеты. Если есть 100 неразыгранных билетов и один из процессов обладает 20 из них, то он будет иметь 20%-ную вероятность выигрыша в каждой лотерее.

Лотерейное планирование

Взаимодействующие процессы могут по желанию обмениваться билетами. Например, когда клиентский процесс отправляет сообщение серверному процессу, а затем блокируется, он может передать все свои билеты серверному процессу, чтобы повысить его шансы быть запущенным следующим.

Справедливое планирование

Если пользователь 1 запускает 9 процессов, а пользователь 2 запускает 1 процесс, то при циклическом планировании или при равных приоритетах пользователь 1 получит 90% процессорного времени, а пользователь 2 получит только 10%

Справедливое планирование

Чтобы избежать подобной ситуации некоторые системы перед планированием работы процесса берут в расчет, кто является его владельцем.

Если каждому из двух пользователей было обещано по 50% процессорного времени, то они его получат, независимо от количества имеющихся у них процессов.

Справедливое планирование

Каждому из пользователей обещано 50% процессорного времени. У первого пользователя четыре процесса: A, B, C и D, а у второго пользователя только один процесс — E.

AEBECEDEAEBECEDE...