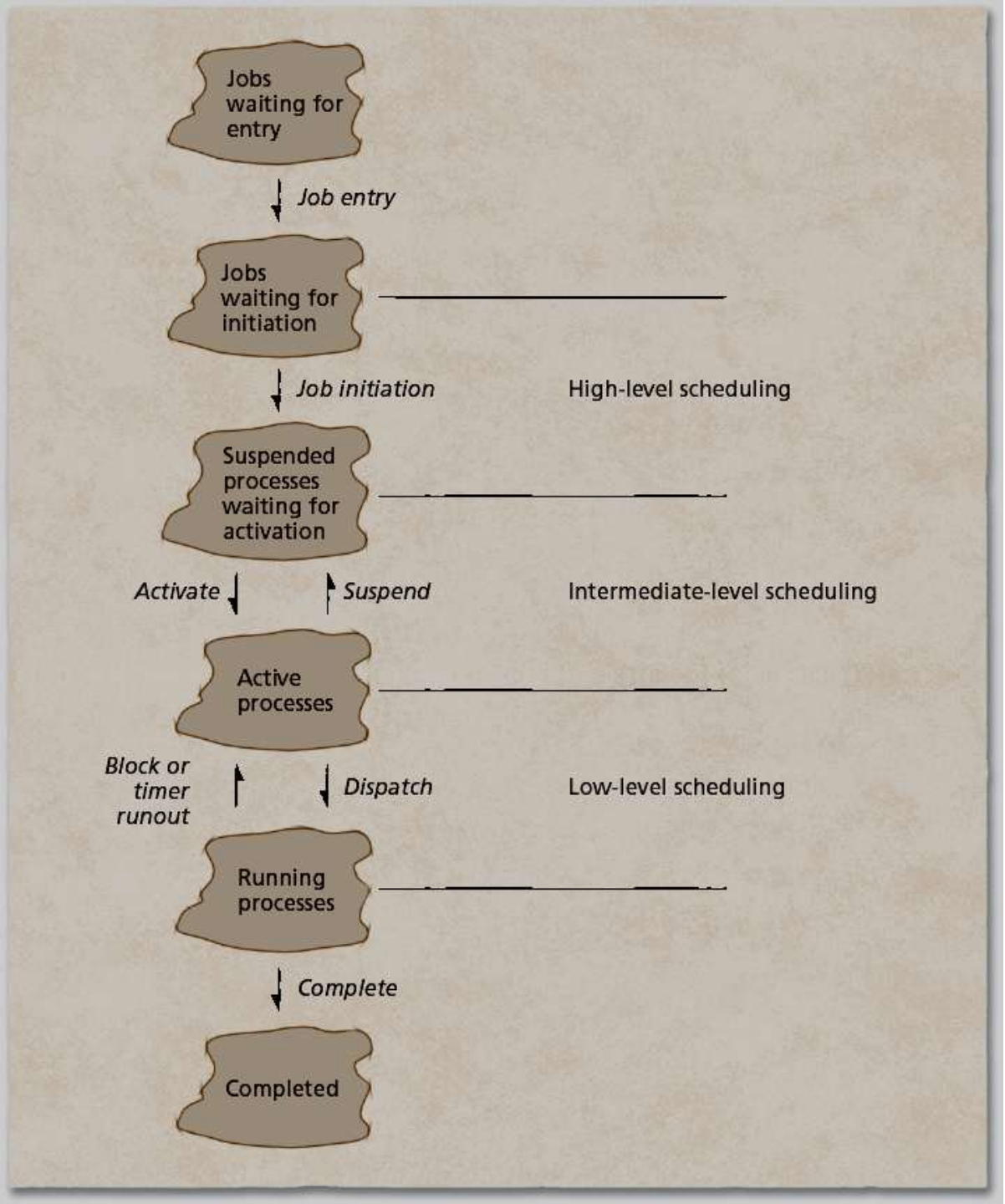
Планирование работы процессора

Уровни планирования



Задания, ожидающие ввода

Задания, ожидающие запуска

Приостановленные процессы, ожидающие активации

Активные процессы

Выполняющиеся процессы

Завершенные процессы

Планирование на верхнем уровне

Опр. Планирование на верхнем уровне (high-level scheduling) – определение заданий, которые могут быть допущены в систему активного участия в соревновании за системные ресурсы.

Планирование на верхнем уровне

- Используется в крупных мэйнфреймах, предназначенных для пакетной обработки данных
- Определяет степень многозадачности

Степень многозадачности

Опр. Степень многозадачности (degree of multiprogramming) – общее количество процессов в системе в конкретный момент времени.

Планирование на промежуточном уровне

Опр. Планирование на промежуточном уровне (intermediate-level scheduling) – на этом уровне планирования определяется, какие процессы будут допущены к планировщику на нижнем уровне.

Планирование на промежуточном уровне

- В большинстве современных систем планировщик верхнего уровня отсутствует, и запуск заданий осуществляется планировщиком промежуточного уровня
- В роли планировщика промежуточного уровня выступает менеджер памяти

Планирование на нижнем уровне

Опр. Планирование на нижнем уровне (low-level scheduling) – определение процесса, который должен получить управление процессором. Осуществляется планировщиком процессов.

Вопрос для самопроверки

- Может ли менеджер памяти запрещать процессам доступ к планировщику процессов? (Да/Нет)

Вопрос для самопроверки

- Может ли менеджер памяти запрещать процессам доступ к планировщику процессов? (Да/Нет)
- Да. Планировщик промежуточного уровня может запрещать процессам доступ к планировщику нижнего уровня, если система перегружена.

Вопрос для самопроверки

- Должен ли планировщик процессов оставаться резидентным в основной памяти? (Да/Нет)

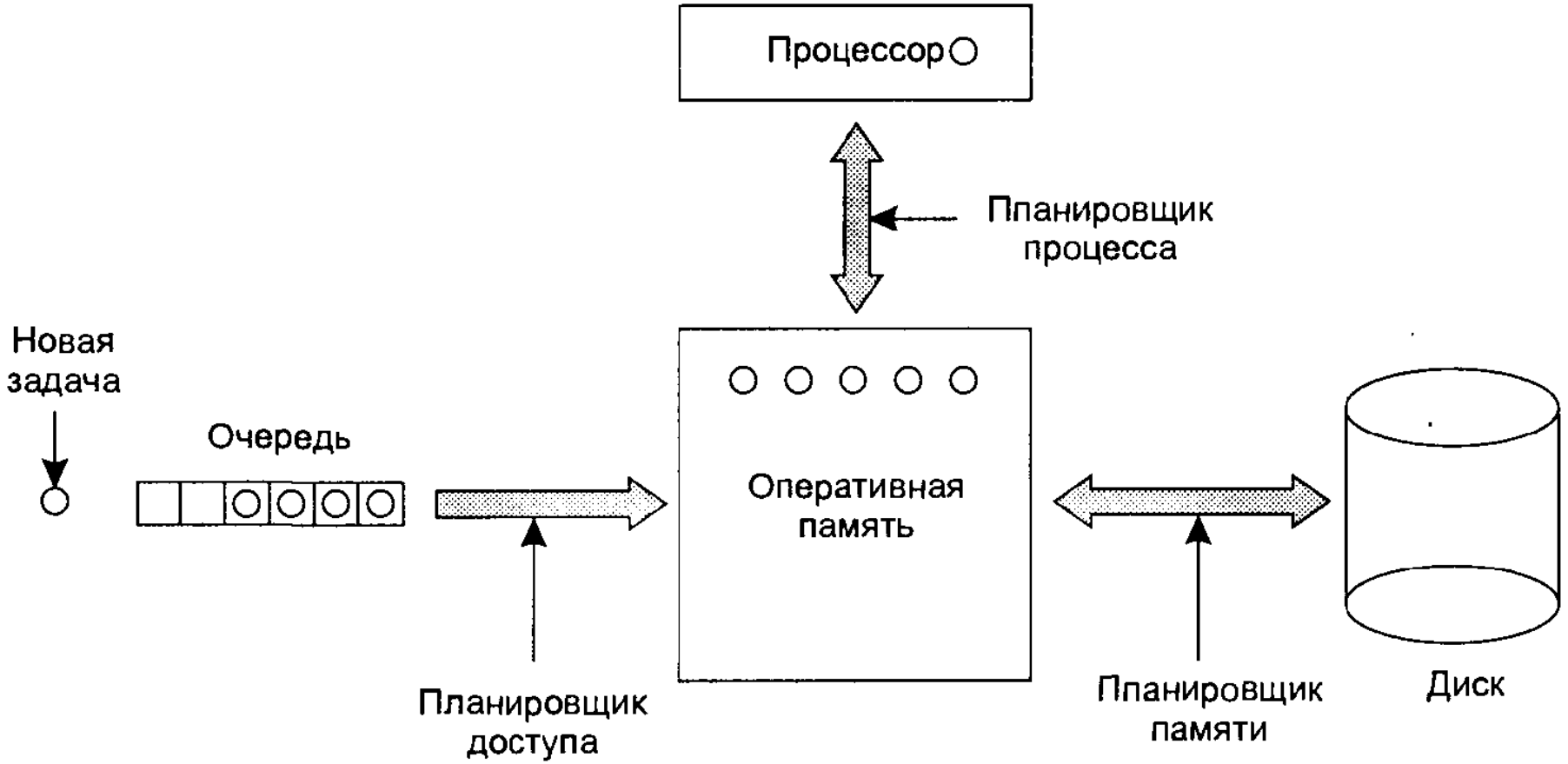
Вопрос для самопроверки

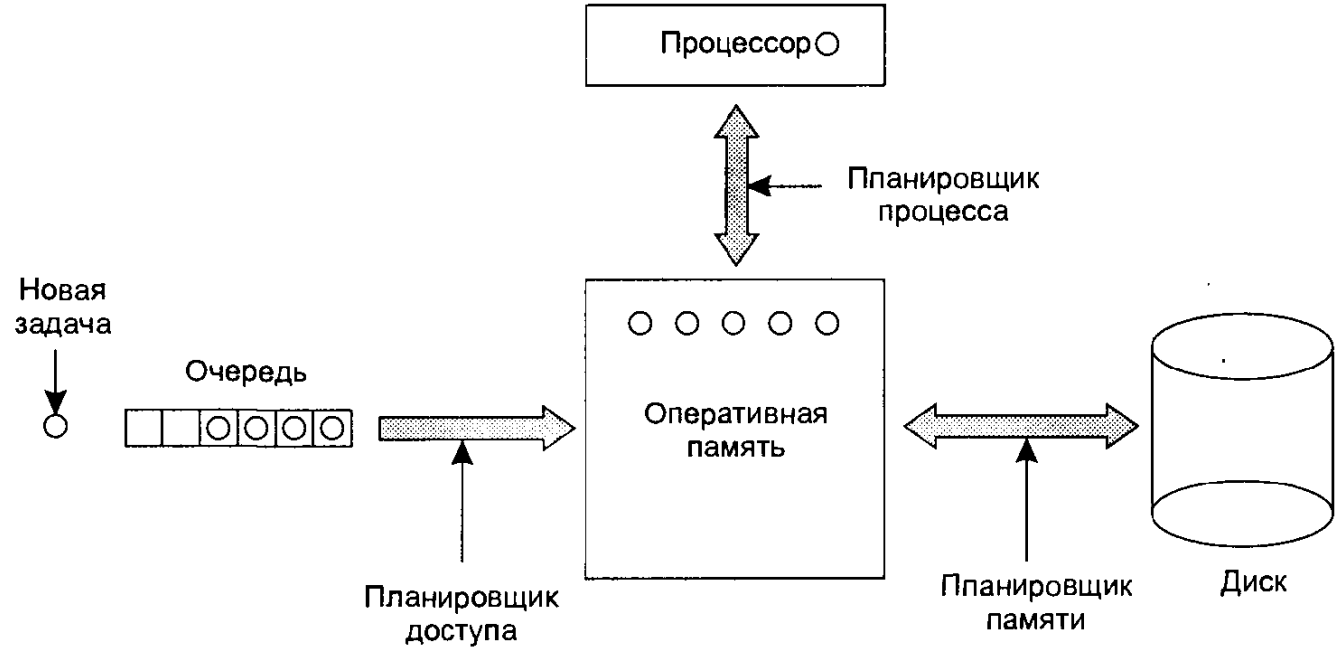
- Должен ли планировщик процессов оставаться резидентным в основной памяти? (Да/Нет)
- Да. Планировщик нижнего уровня должен оставаться резидентным в основной памяти, поскольку код планировщика нужно выполнять достаточно часто, чтобы оперативно реагировать для уменьшения накладных расходов, связанных с планированием выполнения процессов.

Планирование работы процессора

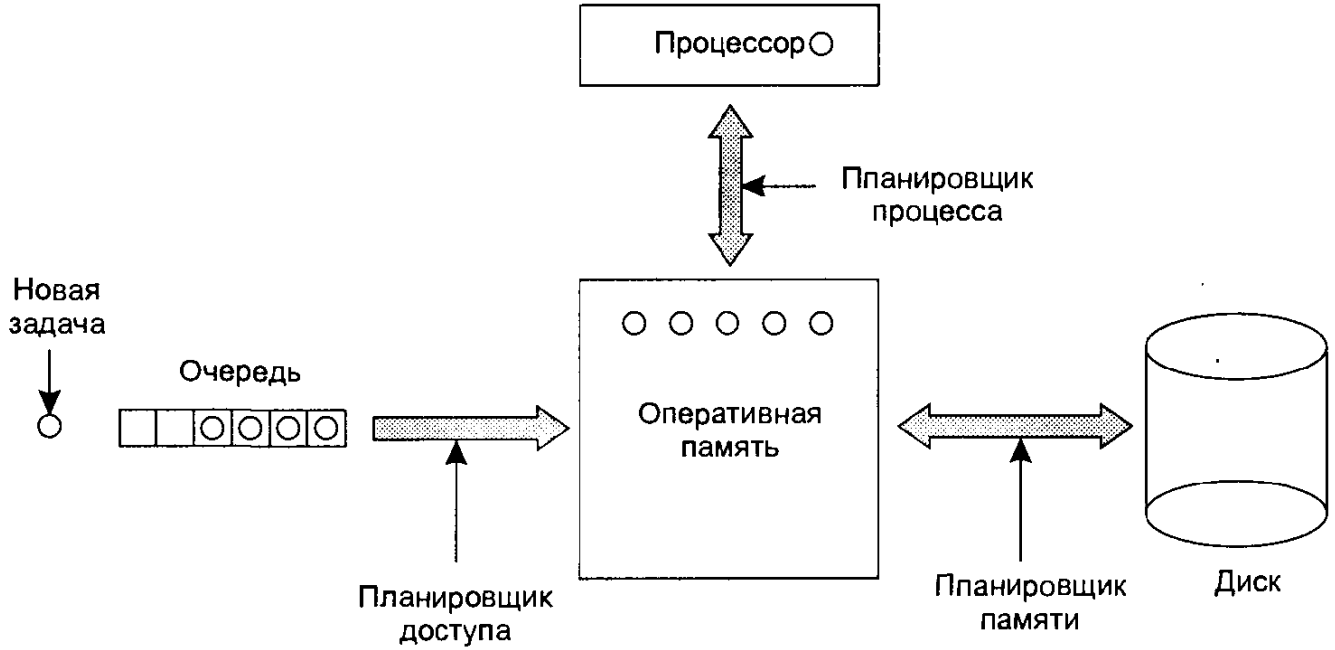
Планирование с приостановкой процессов

Трехуровневое планирование

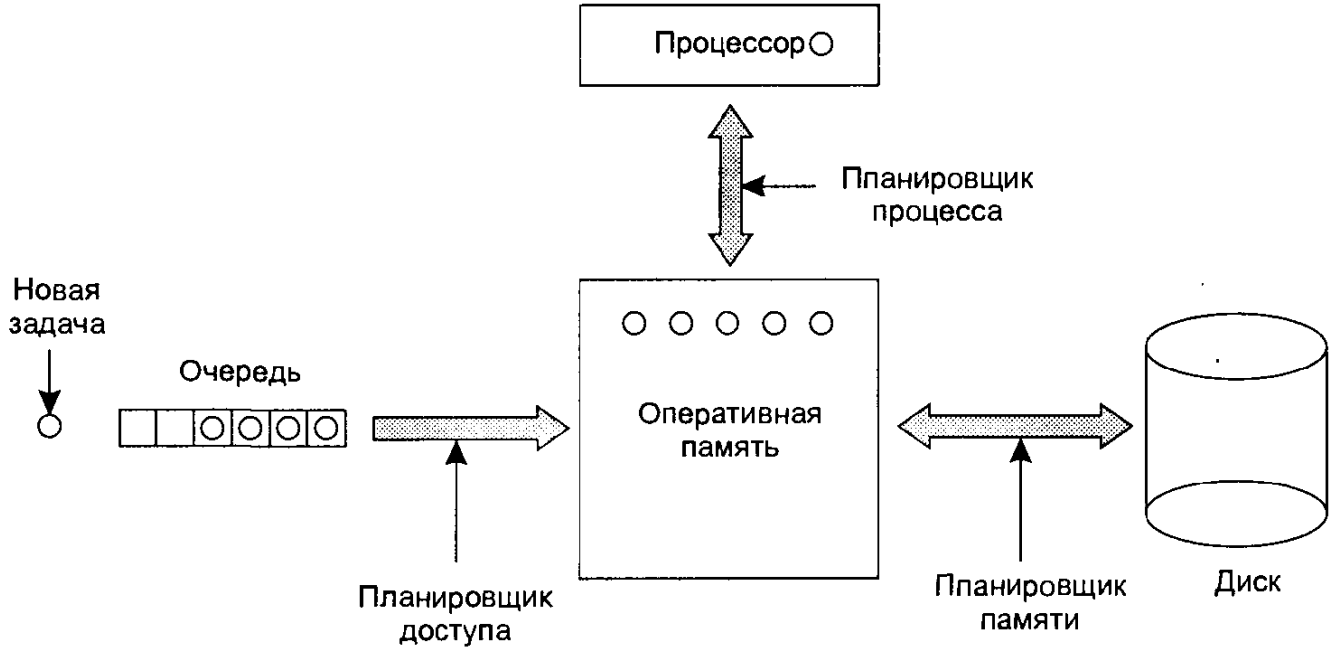




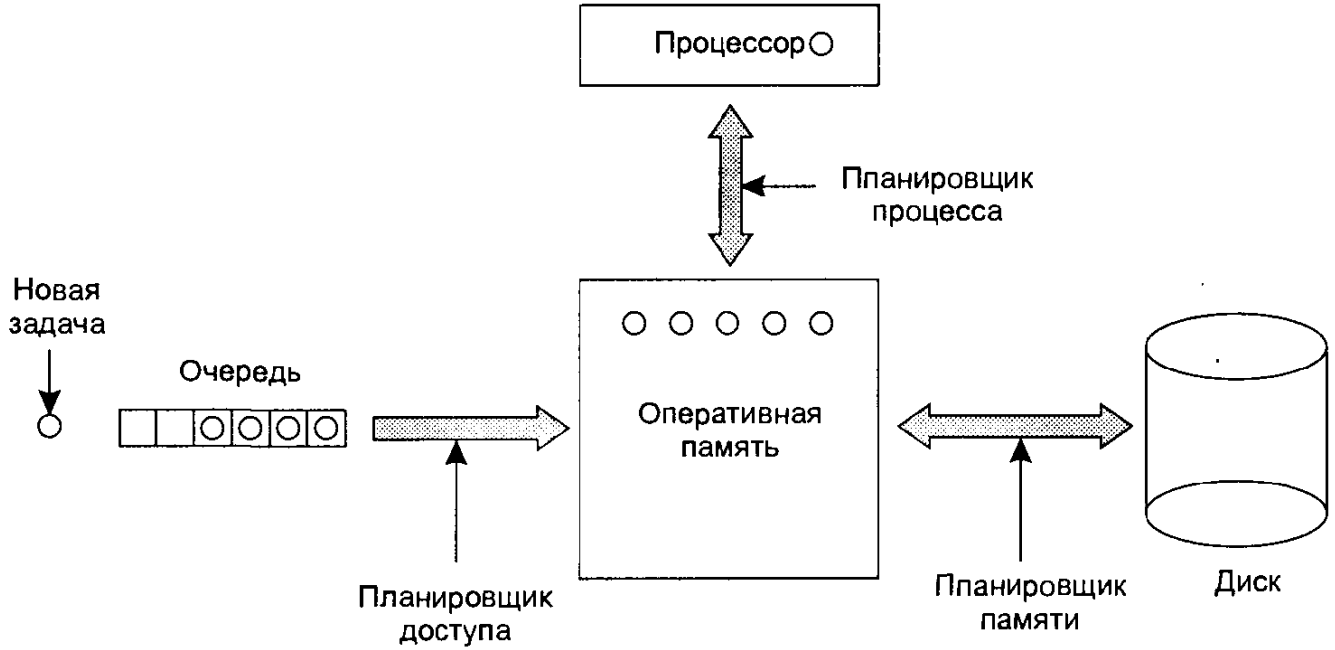
Опр. Активный процесс (active process) – это процесс, находящийся в оперативной памяти и имеющий возможность вести борьбу за процессорное время. Он может находиться в одном из активных состояний (выполняется, готов, блокирован)



Опр. Приостановленный процесс (suspended process) – это процесс, находящийся на диске, на некоторое время выбывший из борьбы за процессорное время, но не прекративший своего существования.



Опр. Приостановка процессов (suspend) – операция, осуществляемая менеджером памяти. Он выгружает процессы на диск, если система перегружена, либо их в памяти слишком много и все они в ней не помещаются.



Опр. Активация процессов (activate) – операция, осуществляемая менеджером памяти. Он периодически просматривает процессы, находящиеся на диске, чтобы решить, какой из них переместить в память.

Критерии активации процессов

- Сколько времени прошло с тех пор, как процесс был выгружен на диск
- Сколько времени процесс уже использовал процессор
- Какова важность процесса
- …

Вопрос для самопроверки

- Может ли планировщик доступа переместить процесс из оперативной памяти в очередь задач? (Да/Нет)

Вопрос для самопроверки

- Может ли планировщик доступа переместить процесс из оперативной памяти в очередь задач? (Да/Нет)
- Нет. Планировщик доступа осуществляет планирование на верхнем уровне и отвечает за допуск задач к активному соревнованию за процессорное время. Он не может выгружать процессы обратно в очередь задач.

Вопрос для самопроверки

- Верно ли, что планировщик процессов вызывается чаще, чем менеджер памяти? (Да/Нет)

Вопрос для самопроверки

- Верно ли, что планировщик процессов вызывается чаще, чем менеджер памяти? (Да/Нет)
- Да. Планировщик процессов отвечает за большинство переходов между активными состояниями процессов, что осуществляется чаще, чем приостановка и активация процессов.

Планирование работы процессора

Планирование с приоритетным вытеснением

Планирование без приоритетного вытеснения

Опр. Планирование без приоритетного вытеснения (nonpreemptive scheduling) – политика планирования, которая не позволяет системе отбирать процессор у процесса до тех пор, пока сам процесс не отдаст его добровольно, либо не закончит работу. Используется, например, в системах пакетной обработки данных.

Планирование с приоритетным вытеснением

Опр. Планирование с приоритетным вытеснением (preemptive scheduling) – политика планирования, позволяющая отбирать процессор у процесса, и основывающаяся на приоритетах процессов. Используется, например, в системах реального времени и интерактивных системах разделения времени.

Приоритет

Опр. Приоритет (priority) – мера важности процесса (потока), используемая для определения порядка и продолжительности его выполнения.

Статические приоритеты

Опр. Статические приоритеты (static priorities) – не изменяющиеся во времени приоритеты. Они присваиваются процессам до начала их выполнения и остаются постоянными.

Динамические приоритеты

Опр. Динамические приоритеты (dynamic priorities) – изменяющиеся во время выполнения процессов приоритеты. Их значение может меняться в зависимости от изменения ситуации.

Вопрос для самопроверки

- Может ли планирование без приоритетного вытеснения иметь преимущество перед планированием с вытеснением? (Да/Нет)

Вопрос для самопроверки

- Может ли планирование без приоритетного вытеснения иметь преимущество перед планированием с вытеснением? (Да/Нет)
- Да. Планирование без приоритетного вытеснения обеспечивает предсказуемое время обработки заданий, что необходимо для систем пакетной обработки данных.

Вопрос для самопроверки

- Может ли процесс, вошедший в бесконечный цикл выполнения, монопольно захватить систему с приоритетным вытеснением? (Да/Нет)

Вопрос для самопроверки

- Может ли процесс, вошедший в бесконечный цикл выполнения, монопольно захватить систему с приоритетным вытеснением? (Да/Нет)
- Да. Если этот процесс имеет самый высокий приоритет. Операционная система может справляться с такими ситуациями, ограничивая максимальное время, в течение которого конкретный процесс может использовать процессор.

Вопрос для самопроверки

- Может ли динамический планировщик отдать предпочтение процессу с низким приоритетом, запросившим недоиспользованный ресурс? (Да/Нет)

Вопрос для самопроверки

- Может ли динамический планировщик отдать предпочтение процессу с низким приоритетом, запросившим недоиспользованный ресурс? (Да/Нет)
- Да. Высока вероятность того, что этот ресурс окажется свободным и процесс, запросивший его сможет его использовать и завершить свою работу еще до того, как высокоприоритетный процесс дождется освобождения занятого ресурса.

Планирование работы процессора

Цели планирования

Цели планирования

1. Обеспечить максимальную пропускную способность системы (число процессов в единицу времени)
2. Гарантировать максимальному числу пользователей, работающих в интерактивном режиме приемлемые времена ответа
3. Обеспечить максимальное использование ресурсов системы

Цели планирования

4. Исключить бесконечное откладывание (когда процесс ожидает появления события, которое может никогда не произойти)
5. Учитывать приоритеты (оказывать предпочтение процессам с более высокими приоритетами)
6. Минимизировать накладные расходы (потеря ресурсов системы на цели планирования)

Цели планирования

7. Обеспечить предсказуемость (гарантировать, что выполнение процесса займет конкретное время независимо от загрузки системы)
8. Обеспечить равноправие (одинаково справедливое отношение ко всем процессам в системе)
9. Обеспечить масштабируемость (плавное снижение производительности при увеличении загрузки системы)

Вопрос для самопроверки

- Противоречит ли цель обеспечить равноправие процессов необходимости учитывать приоритеты? (Да/Нет)

Вопрос для самопроверки

- Противоречит ли цель обеспечить равноправие процессов необходимости учитывать приоритеты? (Да/Нет)
- Да. Трудно обеспечить одинаково справедливое отношение ко всем процессам в системе и одновременно оказывать предпочтение процессам с более высокими приоритетами.

Вопрос для самопроверки

- Противоречит ли цель обеспечить предсказуемость необходимости учитывать приоритеты? (Да/Нет)

Вопрос для самопроверки

- Противоречит ли цель обеспечить предсказуемость необходимости учитывать приоритеты? (Да/Нет)
- Да. Трудно гарантировать, что выполнение процесса займет конкретное время независимо от загрузки системы, если в систему все время будут поступать новые более высокоприоритетные процессы.

Вопрос для самопроверки

- Всегда ли издержки планирования представляют собой бесполезную трату ресурсов? (Да/Нет)

Вопрос для самопроверки

- Всегда ли издержки планирования представляют собой бесполезную трату ресурсов? (Да/Нет)
- Нет. Потеря ресурсов системы на цели планирования при эффективном планировании может с лихвой компенсироваться обеспечением максимального использования процессами этих ресурсов.

Планирование работы процессора

Типы процессов

Процесс, интенсивно использующий ввод/вывод

Опр. Процесс, интенсивно использующий ввод/вывод (I/O-bound) – процесс, который обычно кратковременно использует процессор перед генерацией запроса ввода/вывода.

Процесс, интенсивно использующий процессор

Опр. Процесс, интенсивно использующий процессор (processor-bound) – процесс, расходующий в ходе выполнения весь отведенный ему квант времени. Подобные процессы обычно интенсивно проводят вычисления и генерируют мало запросов ввода/вывода.

Пакетный процесс

Опр. Пакетный процесс (batch process) – процесс, который может работать без вмешательства пользователя.

Интерактивный процесс

Опр. Интерактивный процесс (interactive process) – процесс, которому во время работы необходимо вести диалог с пользователем.

Процесс, выполняемый в режиме реального времени

Опр. Процесс, выполняемый в режиме реального времени (real-time process) – процесс, для которого не приемлемо замедление обслуживания во времени.

Пр. Приложения мультимедиа.

Вопрос для самопроверки

- Пакетные процессы всегда относятся к разновидности процессов, интенсивно использующих процессор? (Да/Нет)

Вопрос для самопроверки

- Пакетные процессы всегда относятся к разновидности процессов, интенсивно использующих процессор? (Да/Нет)
- Нет. Пакетные процессы не взаимодействуют с пользователем, и поэтому их обычно относят к процессам, интенсивно использующим процессор. Если же подобный процесс часто обращается к диску, то его относят к интенсивно использующим ввод/вывод.

Вопрос для самопроверки

- Интерактивные процессы обычно относятся к разновидности процессов, интенсивно использующих ввод/вывод? (Да/Нет)

Вопрос для самопроверки

- Интерактивные процессы обычно относятся к разновидности процессов, интенсивно использующих ввод/вывод? (Да/Нет)
- Да. Интерактивные процессы часто вынуждены ожидать ввода данных от пользователя, поэтому в первую очередь их можно отнести к процессам, интенсивно использующим ввод/вывод.

Вопрос для самопроверки

- Известно ли обычно планировщику, сколько времени процессу осталось до завершения? (Да/Нет)

Вопрос для самопроверки

- Известно ли обычно планировщику, сколько времени процессу осталось до завершения? (Да/Нет)
- Нет. Планировщику редко бывает точно известно, сколько времени необходимо каждому процессу до завершения. Процессы, если им разрешить это, могут неправильно указывать ожидаемое время завершения, чтобы получить лучшие условия обслуживания планировщиком.

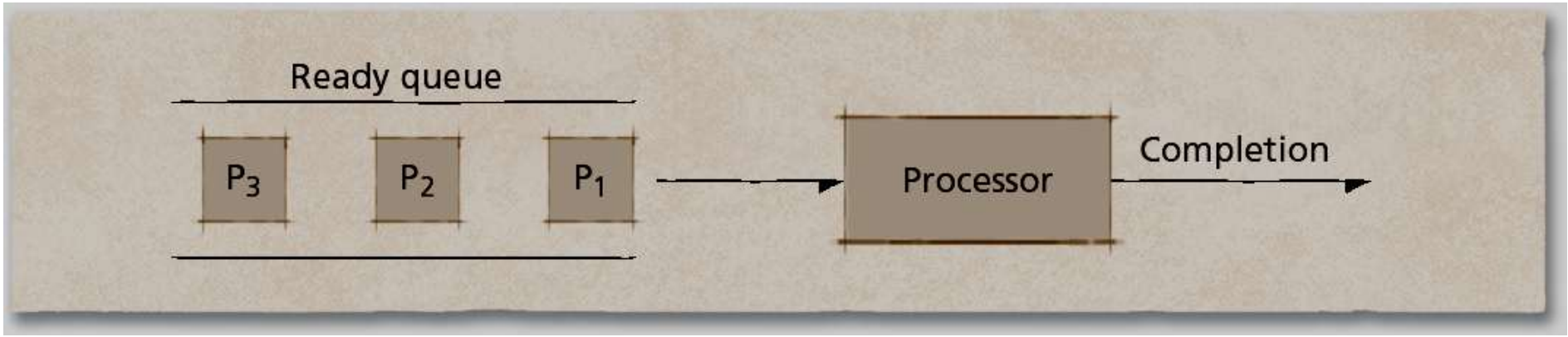
Планирование работы процессора

Базовые алгоритмы планирования: планирование по принципу FIFO

Планирование по принципу FIFO

Опр. Планирование по принципу FIFO (first in – first out, первым пришел – первым ушел) – механизм планирования, согласно которому процессор поступает в распоряжение процессов в порядке их попадания в очередь готовых процессов. FIFO не поддерживает приоритетное вытеснение – процесс продолжает свое выполнение до полного завершения.

Планирование по принципу FIFO



Очередь готовности **Завершение**

Планирование по принципу FIFO

- Задает простейший план выполнения процессов
- Менее важные процессы могут заставить ждать своей очереди более важные для системы процессы
- В современных системах редко используется самостоятельно в качестве основной дисциплины обслуживания

Планирование по принципу FIFO

- Можно встретить во многих современных алгоритмах планирования (например, в алгоритме планирования на основе приоритетов, где процессы с одинаковым приоритетом обслуживаются по принципу FIFO)

Вопрос для самопроверки

- Может ли в системе, использующей принцип FIFO, возникнуть ситуация бесконечного откладывания, если все процессы должны обязательно завершить свою работу? (Да/Нет)

Вопрос для самопроверки

- Может ли в системе, использующей принцип FIFO, возникнуть ситуация бесконечного откладывания, если все процессы должны обязательно завершить свою работу? (Да/Нет)
- Нет. Бесконечное откладывание не должно возникнуть, поскольку все пребывающие процессы будут помещены в конец очереди, не мешая выполняться ждущим процессам.

Вопрос для самопроверки?

- Правда ли, что в современных системах принцип планирования FIFO используется редко? (Да/Нет)

Вопрос для самопроверки?

- Правда ли, что в современных системах принцип планирования FIFO используется редко? (Да/Нет)
- Нет. Его можно встретить во многих современных алгоритмах планирования в качестве составной части более сложных алгоритмов.

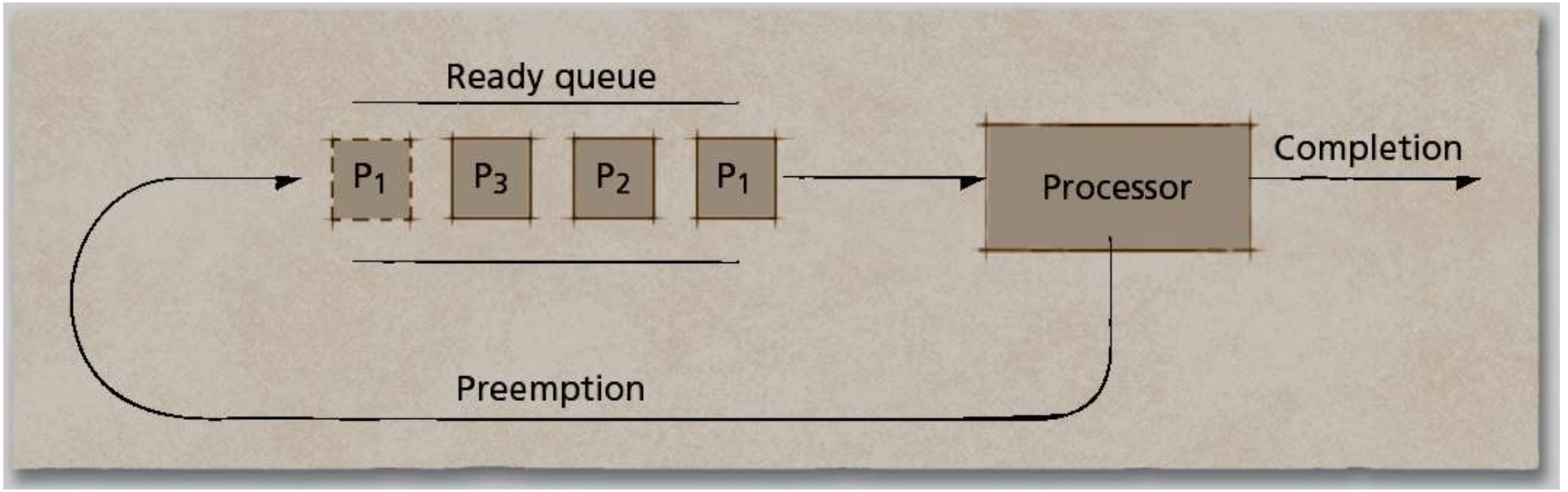
Планирование работы процессора

Базовые алгоритмы планирования: циклическое планирование (RR)

Циклическое планирование (RR)

Опр. Циклическое планирование (round-robin (RR) scheduling) – политика планирования, согласно которой каждый процесс в состоянии готовности может выполняться в течение не более, чем одного кванта в каждом цикле. После того как по одному разу будет запущен каждый процесс из очереди, планировщик начнет новый цикл, запустив на выполнение первый процесс из очереди.

Циклическое планирование (RR)



Приоритетное вытеснение

Циклическое планирование (RR)

- Эффективно для работы в интерактивной среде
- В современных системах редко используется самостоятельно в качестве основной дисциплины обслуживания
- Можно встретить во многих современных алгоритмах планирования (например, в системах реального времени)

Вопрос для самопроверки

- Может ли дисциплина планирования RR учитывать приоритеты процессов? (Да/Нет)

Вопрос для самопроверки

- Может ли дисциплина планирования RR учитывать приоритеты процессов? (Да/Нет)
- Да. Для этого надо изменять величину кванта времени, и процессам разных приоритетов выделять кванты разных размеров.

Вопрос для самопроверки

- Верно ли, что алгоритм RR является менее приемлемым для обслуживания интерактивных пользователей, чем принцип FIFO? (Да/Нет)

Вопрос для самопроверки

- Верно ли, что алгоритм RR является менее приемлемым для обслуживания интерактивных пользователей, чем принцип FIFO? (Да/Нет)
- Нет. Планирование по принципу FIFO заставляет ждать завершения работы выполняющегося процесса, что может оказаться неприемлемым для интерактивных процессов.

Планирование работы процессора

Величина кванта времени

Квант

Опр. Квант (quantum) – временной интервал, в течение которого процесс может использовать процессор до момента приоритетного вытеснения.

Оптимальная величина кванта

- Меняется от системы к системе, от процесса к процессу
- Зависит от нагрузок
- Для процессов интенсивно использующих ввод/вывод равна времени, необходимому каждому процессу, чтобы сгенерировать запрос ввода/вывода, а затем отдать процессор следующему процессу

Оптимальная величина кванта

- Для интерактивных процессов слишком малый квант приведет к частому переключению процессов и небольшой эффективности, слишком большой квант может привести к медленному реагированию на короткие интерактивные нагрузки

Величина квантов в Linux

- По умолчанию для каждого процесса равна 100 мс (1 мс = 10^{-3} с)
- Может меняться от 10 до 200 мс в зависимости от поведения и приоритета процесса

Величина квантов в Windows XP

- По умолчанию для каждого процесса равна в большинстве случаев 20 мс
- Зависит от архитектуры системы
- Может быть разной для разных процессов

Вопрос для самопроверки

- Верно ли, что оптимальная величина кванта времени равна 20 мс? (Да/Нет)

Вопрос для самопроверки

- Верно ли, что оптимальная величина кванта времени равна 20 мс? (Да/Нет)
- Нет. Она может меняться в зависимости от поведения и приоритета процесса.

Вопрос для самопроверки

- Можно ли заранее определить оптимальную величину кванта для процессов интенсивно использующих ввод/вывод? (Да/Нет)

Вопрос для самопроверки

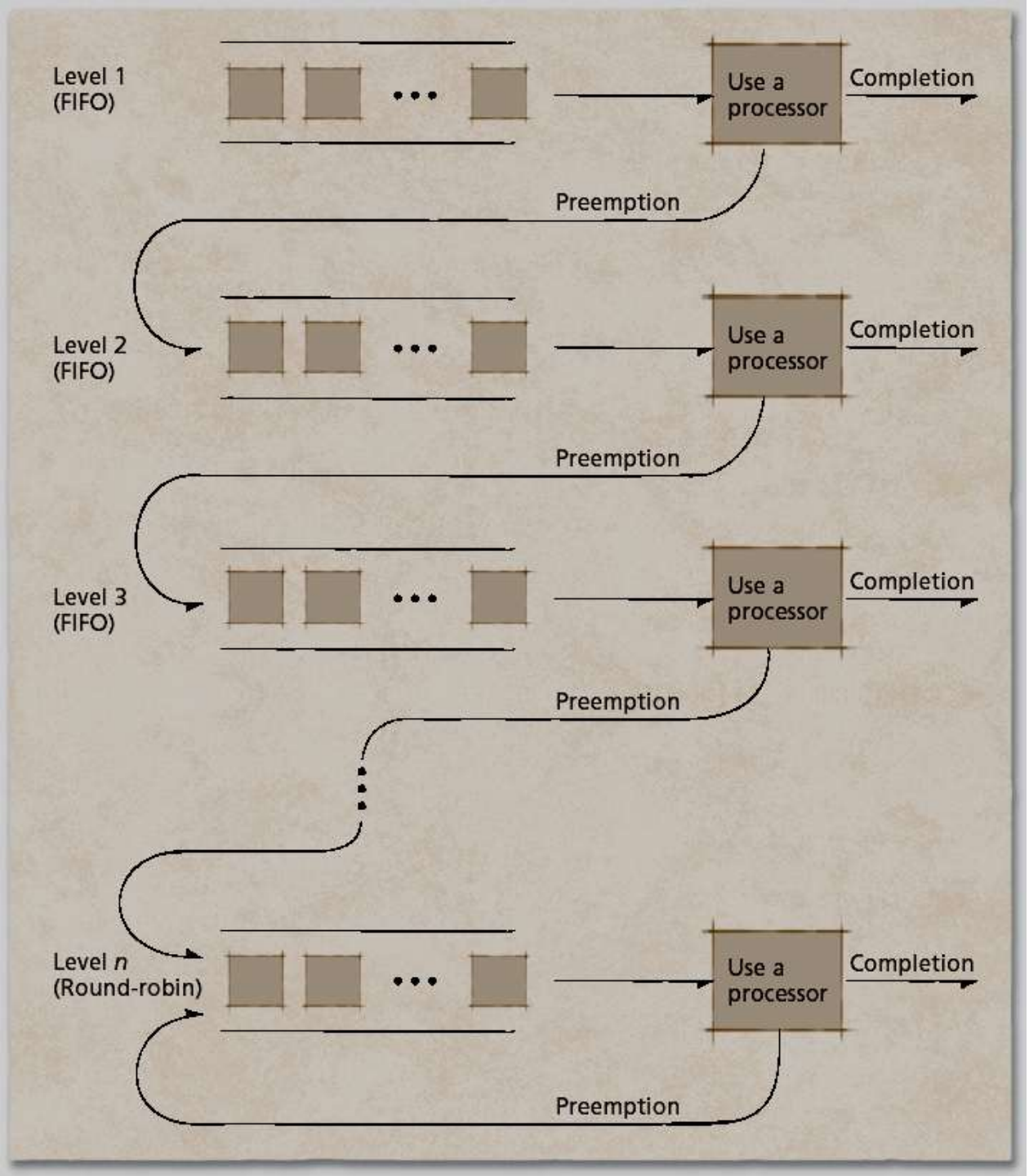
- Можно ли заранее определить оптимальную величину кванта для процессов интенсивно использующих ввод/вывод? (Да/Нет)
- Нет. В общем случае невозможно спрогнозировать ход выполнения программы, то есть система не может точно сказать, когда процесс сгенерирует запрос ввода/вывода.

Планирование работы процессора

Многоуровневые очереди с обратной связью

Многоуровневая очередь с обратной связью

Опр. Многоуровневая очередь с обратной связью (multilevel feedback queue) – структура, обеспечивающая группирование процессов с одинаковым приоритетом в одну очередь, для обслуживания которой применяется принцип FIFO, или алгоритм RR.



Многоуровневая очередь с обратной связью

Базовый алгоритм

1. Новый процесс входит в сеть очередей с конца верхней очереди
2. Процесс перемещается по очереди, реализующей принцип FIFO, пока не получит в свое распоряжение процессор
3. Если процесс завершает свою работу до конца выделенного ему кванта времени, он уходит на завершение

Базовый алгоритм

4. Если до конца выделенного процессу кванта времени он генерирует запрос ввода/вывода, процесс блокируется и выходит из сети очередей
5. После блокирования процесс входит в ту же очередь, из которой он вышел

Базовый алгоритм

4. Если выделенный квант времени истекает до того, как процесс добровольно освободит процессор, этот процесс помещается в конец следующей очереди более низкого уровня с меньшим приоритетом и большим квантом времени

Базовый алгоритм

6. Если данный процесс продолжит использовать полный квант времени, предоставляемый на каждом уровне, он и дальше будет переходить в конец очереди следующего нижележащего уровня
7. В системе предусматривается очередь самого нижнего уровня, которая реализует принцип RR, в которой данный процесс циркулирует до тех пор, пока не закончит свою работу

Многоуровневая очередь с обратной связью

- Процессы, интенсивно использующие процессор, помещаются в очереди более низкого приоритета, поскольку пакетные процессы обычно не требуют малых времен ответа

Многоуровневая очередь с обратной связью

- Процессы, интенсивно использующие ввод/вывод, которые используют процессор кратковременно перед работой с устройствами ввода/вывода, остаются в высоко приоритетных очередях
- В роли подобных процессов обычно выступают интерактивные процессы, которые нуждаются в малых временах ответа

Вопрос для самопроверки?

- Верно ли, что верхняя очередь имеет самый высокий приоритет? (Да/Нет)

Вопрос для самопроверки?

- Верно ли, что верхняя очередь имеет самый высокий приоритет? (Да/Нет)
- Да. Процесс первоначально входит в сеть очередей, начиная с верхней очереди, имеющей самый высокий приоритет.

Вопрос для самопроверки

- Верно ли, что в очередях более низкого уровня процессам выделяются большие кванты времени? (Да/Нет)

Вопрос для самопроверки

- Верно ли, что в очередях более низкого уровня процессам выделяются большие кванты времени? (Да/Нет)
- Да. Чем ниже уровень очереди, тем большие кванты времени выделяются процессам. Интерактивные процессы циркулируют в верхних очередях, интенсивно использующие процессор – в нижних.

Вопрос для самопроверки

- Верно ли, что в самой нижней очереди реализован принцип циклического обслуживания процессов? (Да/Нет)

Вопрос для самопроверки

- Верно ли, что в самой нижней очереди реализован принцип циклического обслуживания процессов? (Да/Нет)
- Да. Это сделано для того, чтобы процессы, дошедшие до самой нижней очереди, в конце концов смогли закончить свою работу.

Планирование работы процессора

Обслуживание процессов разных типов

Обслуживание процессов, интенсивно использующих ввод/вывод

- Таким процессам отдается предпочтение, поскольку они входят в сеть очередей с высоким приоритетом и быстро получают процессор в свое распоряжение
- Квант времени для первой очереди выбирается достаточно большим, с тем чтобы подавляющее большинство процессов, связанных с вводом/выводом (в том числе интерактивных), успели выдать запрос ввода/вывода еще до истечения кванта

Обслуживание процессов, интенсивно использующих ввод/вывод

- Когда подобный процесс выдает запрос ввода/вывода, он блокируется и выходит из сети очередей, получив приоритетное обслуживание
- Повторный вход процесса происходит тогда, когда он снова переходит в состояние готовности
- После блокирования процесс входит в ту же очередь, из которой он вышел

Обслуживание процессов, интенсивно использующих процессор

- Процесс поступает в очередь сети с самым высоким приоритетом
- На этом этапе очередь не знает, к какой категории относится данный процесс – и цель сети выяснить это как можно скорее
- Процесс получает в свое распоряжение процессор весьма быстро, однако после истечения выделенного ему кванта времени процесс переходит в очередь следующего нижележащего уровня

Обслуживание процессов, интенсивно использующих процессор

- Теперь этот процесс будет иметь более низкий приоритет, так что поступающие процессы будут первыми получать в свое распоряжение процессор
- Интерактивные процессы будут продолжать получать хорошие времена ответа, даже если многие процессы интенсивно использующие процессор, постоянно теряют приоритет, переходя на очереди нижележащих уровней

Обслуживание процессов, интенсивно использующих процессор

- В конце концов, процесс, интенсивно использующий процессор, получит его в свое распоряжение
- Ему будет выделен квант времени большей величины, чем в очередях высшего приоритета, но он снова использует этот квант полностью
- Затем он будет перемещен в очередь следующего нижележащего уровня

Обслуживание процессов, интенсивно использующих процессор

- Таким образом, данный процесс будет продолжать переходить в очереди с более низкими приоритетами
- Он будет дольше ждать в промежутках между выделяемыми квантами времени
- Этот процесс будет каждый раз полностью использовать свой квант, когда будет получать в свое распоряжение процессор

Обслуживание процессов, интенсивно использующих процессор

- В конце концов, этот вычислительный процесс окажется в очереди самого низкого уровня с циклическим обслуживанием
- В этой очереди он будет циркулировать пока не завершится

Вопрос для самопроверки

- После блокирования процесс входит в ту же очередь, из которой он вышел, согласно базовому алгоритму многоуровневой очереди? (Да/Нет)

Вопрос для самопроверки

- После блокирования процесс входит в ту же очередь, из которой он вышел, согласно базовому алгоритму многоуровневой очереди? (Да/Нет)
- Да. Когда процесс переходит в состояние готовности после блокирования, он вновь входит в сеть очередей, причем, в ту же очередь из которой он вышел.

Вопрос для самопроверки

- Верно ли, что процесс, максимально интенсивно использующий процессор в конце концов будет обслуживаться в самой нижней очереди? (Да/Нет)

Вопрос для самопроверки

- Верно ли, что процесс, максимально интенсивно использующий процессор в конце концов будет обслуживаться в самой нижней очереди? (Да/Нет)
- Да. Процесс использующий процессор максимальным образом и не выдающий запросов ввода/вывода опустится на самый нижний уровень.

Планирование работы процессора

Обслуживание процессов разных типов: модификации базового алгоритма

- Проблема: Как сразу во время запуска разделять процессы на категории в соответствии с их потребностями в процессорном времени?

- Решение: Когда процесс выходит из сети очередей, он может быть помечен признаком очереди самого низкого уровня, в которой он побывал
- Когда этот процесс в последствии вновь войдет в сеть очередей, он будет направлен прямо в ту очередь, в которой он в последний раз завершил работу

- Проблема: Как тогда реагировать на изменение характера процесса, например, на то, что процесс ранее интенсивно использовавший процессор, начинает преимущественно использовать ввод/вывод?

- Решение: Эту проблему можно решить, если в метке, которой сопровождается процесс, указать также степень использования им выделенных квантов времени во время последнего пребывания в сети очередей

- Если процесс полностью использует выделенный ему квант времени, система поместит его в очередь более низкого уровня
- Если процесс сгенерирует запрос ввода/вывода до того, как закончится квант, он может быть помещен в очередь более высокого уровня

- Проблема: Как учитывать время ожидания обслуживания процесса?

- Решение: Планировщик может использовать механизм старения процессов, помещая процесс в очередь более высокого уровня после того, как процесс провел определенное время, ожидая обслуживания

- Проблема: Как еще более улучшить обслуживание процессов, интенсивно использующих ввод/вывод по сравнению с процессами, интенсивно использующими процессор?

- Решение: Процесс может несколько раз циркулировать в каждой очереди, прежде чем перейти в очередь следующего нижележащего уровня
- Количество подобных циклов может увеличиваться по мере перехода процесса на нижележащие уровни

Вопрос для самопроверки

- Верно ли, что многоуровневая очередь с обратной связью является примером адаптивного механизма? (Да/Нет)

Вопрос для самопроверки

- Верно ли, что многоуровневая очередь с обратной связью является примером адаптивного механизма? (Да/Нет)
- Да. Адаптивный механизм – это механизм, позволяющий реагировать на изменение поведения контролируемой им системы. Многоуровневая очередь с обратной связью является показательным примером адаптивного механизма.

Вопрос для самопроверки

- Верно ли, что процесс всегда входит в сеть очередей с конца верхней очереди? (Да/Нет)

Вопрос для самопроверки

- Верно ли, что процесс всегда входит в сеть очередей с конца верхней очереди? (Да/Нет)
- Нет. Процесс может быть направлен прямо в ту очередь, в которой он в последний раз завершил работу.

Планирование работы процессора

Многоуровневые очереди с обратной связью: оптимальное число очередей и уровней приоритета

- Проблема: Как обеспечить максимальную пропускную способность системы (число процессов в единицу времени) для процессов, интенсивно использующих процессор?

- Решение: При увеличении числа уровней процессы, интенсивно использующие процессор, вынуждены дольше ждать, что приводит к снижению пропускной способности
- Необходимо уменьшить число уровней

- Проблема: Как обеспечить максимальное использование процессорного времени и устройств ввода/вывода?

- Решение: Необходимо увеличить число очередей
- С увеличением числа уровней многим процессам удается завершить работу и сгенерировать запрос ввода/вывода прежде, чем истечет выделенный им квант времени
- Это позволяет эффективно использовать процессорное время и устройства ввода/вывода

Планировщик Linux

- Различает 40 различных уровней приоритета: от -20 до 19
- В соответствии с конвенцией UNIX наименьшее значение означает наибольший приоритет
- -20 – самый высокий приоритет, который может иметь процесс (поток)

Планировщик Linux

- Каждому процессу (потоку) при запуске присваивается приоритет, который может быть динамически изменен
- Приоритет может быть изменен, но не более чем на 5 единиц в любую сторону в зависимости от поведения процесса (потока)

Планировщик Windows XP

- Различает 32 уровня приоритета от 0 до 31
- Потоки реального времени (т.е. такие потоки, которые должны обеспечить быструю реакцию на запросы пользователей) занимают верхние 16 уровней (от 16 до 32)
- Их приоритеты являются статическими и не меняются во время выполнения

Планировщик Windows XP

- Нижние 16 уровней занимают динамические потоки, их приоритеты могут меняться в зависимости от поведения
- Однако динамический приоритет потока не может опуститься ниже определенного для него базового уровня, либо подняться до диапазона приоритетов реального времени

Вопрос для самопроверки

- Для обеспечения максимального использования вычислительных ресурсов необходимо уменьшать число очередей? (Да/Нет)

Вопрос для самопроверки

- Для обеспечения максимального использования вычислительных ресурсов необходимо уменьшать число очередей? (Да/Нет)
- Нет. Необходимо увеличить число очередей. Чем больше очередей, тем точнее учитываются потребности процессов в ресурсах.

Вопрос для самопроверки

- Для ускорения обслуживания процессов, интенсивно использующих процессор, необходимо уменьшать число очередей? (Да/Нет)

Вопрос для самопроверки

- Для ускорения обслуживания процессов, интенсивно использующих процессор, необходимо уменьшать число очередей? (Да/Нет)
- Да. Необходимо уменьшить число уровней. При увеличении числа уровней процессы, интенсивно использующие процессор, вынуждены дольше ждать, что приводит к снижению пропускной способности.

Вопрос для самопроверки

- Планировщики Linux и Windows XP различают одинаковое число уровней приоритета? (Да/Нет)

Вопрос для самопроверки

- Планировщики Linux и Windows XP различают одинаковое число уровней приоритета? (Да/Нет)
- Нет. Планировщики Linux различает 40 различных уровней приоритета, а Windows XP – 32.

Планирование работы процессора

Планирование потоков Java

Планирование потоков Java

- Любой Java-апплет является многопоточным
- Приоритет любого потока Java может находиться в пределах от **`Thread.MIN_PRIORITY=1`** до **`Thread.MAX_PRIORITY=10`**
- По умолчанию каждому потоку присваивается приоритет **`Thread.NORM_PRIORITY=5`**

Планирование потоков Java

- Каждый новый поток получает приоритет, равный приоритету породившего его потока
- Если потоки реализуются в пространстве пользователя, Java-библиотеки времени выполнения используют квантование времени

Квантование времени

Опр. Квантование времени (timeslicing) – задание для потоков плана выполнения, согласно которому каждый поток может выполняться в течение не более чем одного кванта до момента приоритетного вытеснения.

Поток уровня пользователя

Опр. Поток уровня пользователя (user-level thread) – модель потоков, согласно которой все потоки процесса связываются с одним контекстом выполнения.

Потоки уровня пользователя

- При задании плана выполнения многопоточных процессов, реализованных в виде потоков пользовательского уровня, операционной системе ничего не известно о количестве потоков процесса
- Операционная система оперирует ими как единым блоком, требуя наличия библиотеки уровня пользователя для планирования выполнения потоков, входящих в состав данного процесса

Поток уровня ядра

Опр. Поток уровня ядра (kernel-level thread) – поток, создаваемый самой операционной системой.

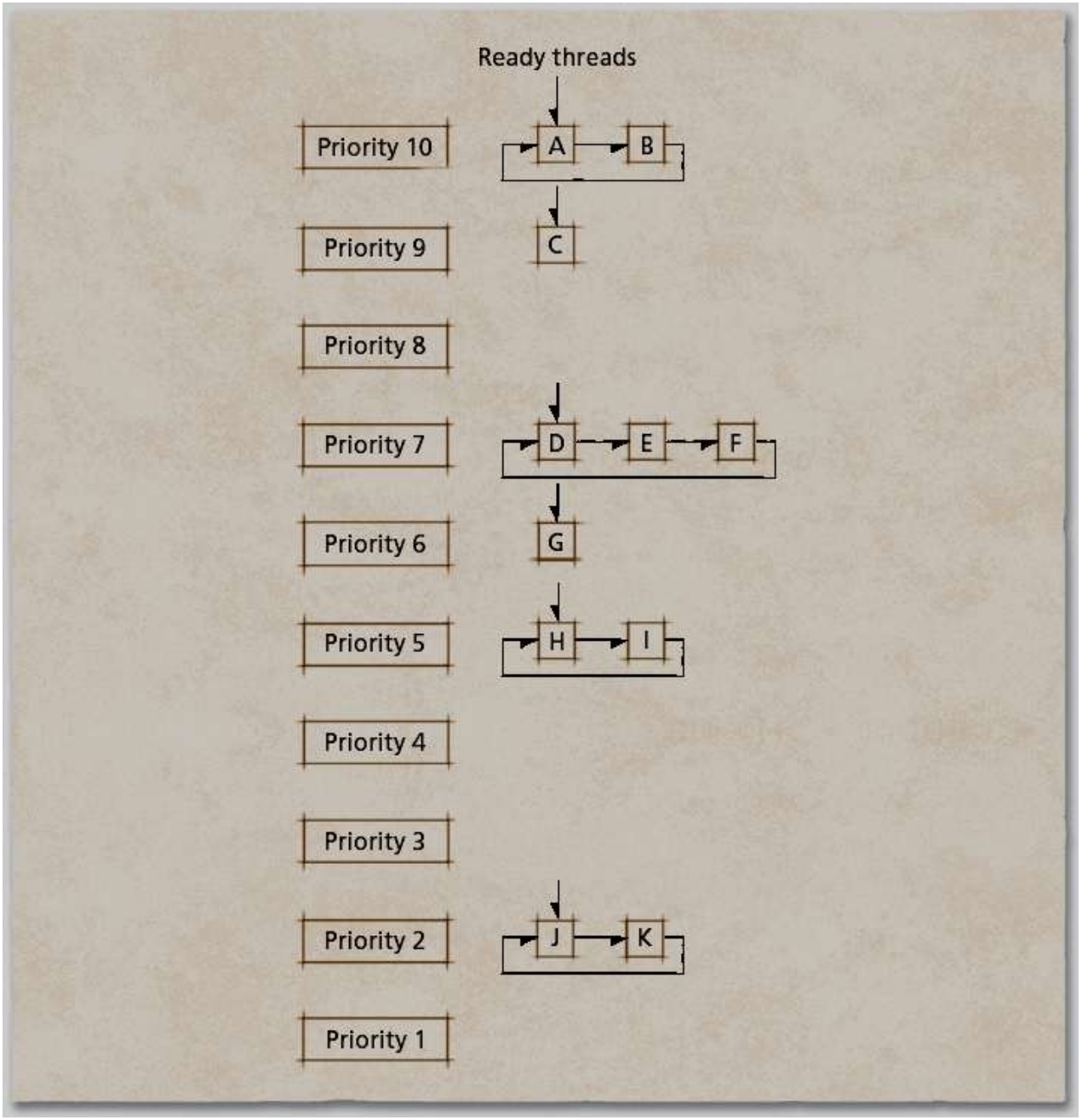
Потоки уровня ядра

- Если система поддерживает потоки уровня ядра, она может задавать план выполнения каждого потока независимо от других потоков этого процесса
- В зависимости от платформы, потоки Java могут быть реализованы
  - В пространстве пользователя (некоторые ранние версии UNIX и OS Solaris)
  - В пространстве ядра (Linux, Windows XP)

Планирование потоков Java

- Планировщик потоков Java гарантирует, что поток с самым высоким приоритетом в виртуальной машине Java будет выполняться всегда
- Если в системе существует несколько потоков с одинаковым приоритетом, то для их обслуживания будет использована дисциплина RR

План выполнения потоков Java по приоритетам



1. Потоки A и B выполняются согласно алгоритму RR до своего окончания

2. До самого конца выполняется поток C

3. И т.д.

Метод `yield` класса `Thread`

- Позволяет текущему потоку добровольно освобождать процессор, чтобы позволить выполняться потоку с таким же приоритетом
- Необходимость в функции `yield` существует только для систем, в которых не поддерживается квантование времени

Метод **`yield`** класса **`Thread`**

- В таких системах поток обычно выполняется до завершения, прежде чем другой поток с таким же приоритетом получит возможность продолжить выполнение

Метод **`yield`** класса **`Thread`**

- Поскольку приложения Java нацелены на переносимость между платформами и поскольку программисты не всегда знают, что конкретная платформа поддерживает квантование времени, программисты применяют метод **`yield`**, чтобы гарантировать корректную работу своего приложения

Вопрос для самопроверки

- Могут ли прибывающие потоки более высокого приоритета привести к бесконечному откладыванию потоков с более низким приоритетом? (Да/Нет)

Вопрос для самопроверки

- Могут ли прибывающие потоки более высокого приоритета привести к бесконечному откладыванию потоков с более низким приоритетом? (Да/Нет)
- Да. Эта ситуация может произойти с потоками Java, если в операционной системе не будет предусмотрена соответствующая защита.

Вопрос для самопроверки

- Поток Java с более низким приоритетом никогда не сможет начать работу, пока поток более высокого приоритета находится в состоянии готовности? (Да/Нет)

Вопрос для самопроверки

- Поток Java с более низким приоритетом никогда не сможет начать работу, пока поток более высокого приоритета находится в состоянии готовности? (Да/Нет)
- Да. Механизм планирования Java будет выполнять поток с самым высоким приоритетом, находящийся в состоянии готовности.