ПЛАНИРОВАНИЕ

Планировщик – это часть операционной системы, выбирающая между активными процессами, желающими получить доступ к процессору

Алгоритмом планирования – алгоритм, в соответствии с которым процессор осуществляет этот выбор

Планировщик также должен заботиться об эффективном использовании процессора, поскольку переключение между процессами требует затрат:

1. Необходимо переключиться из режима пользователя в режим ядра.
2. Сохранить состояние текущего процесса, включая сохранение регистров в таблице процессов, чтобы их можно было загрузить заново позже.
3. Нужно выбрать следующий процесс, запустив алгоритм планирования.

перезапустить блок управления памятью с картой памяти нового

Поведение процесса

Практически все процессы чередуют периоды вычислений с операциями (дисковыми) ввода-вывода. Обычно процессор некоторое время работает без остановки, затем происходит системный вызов на чтение из файла или запись в файл. После выполнения системного вызова процессор опять считает, пока ему не понадобятся новые данные или не потребуется записать полученные данные и т.

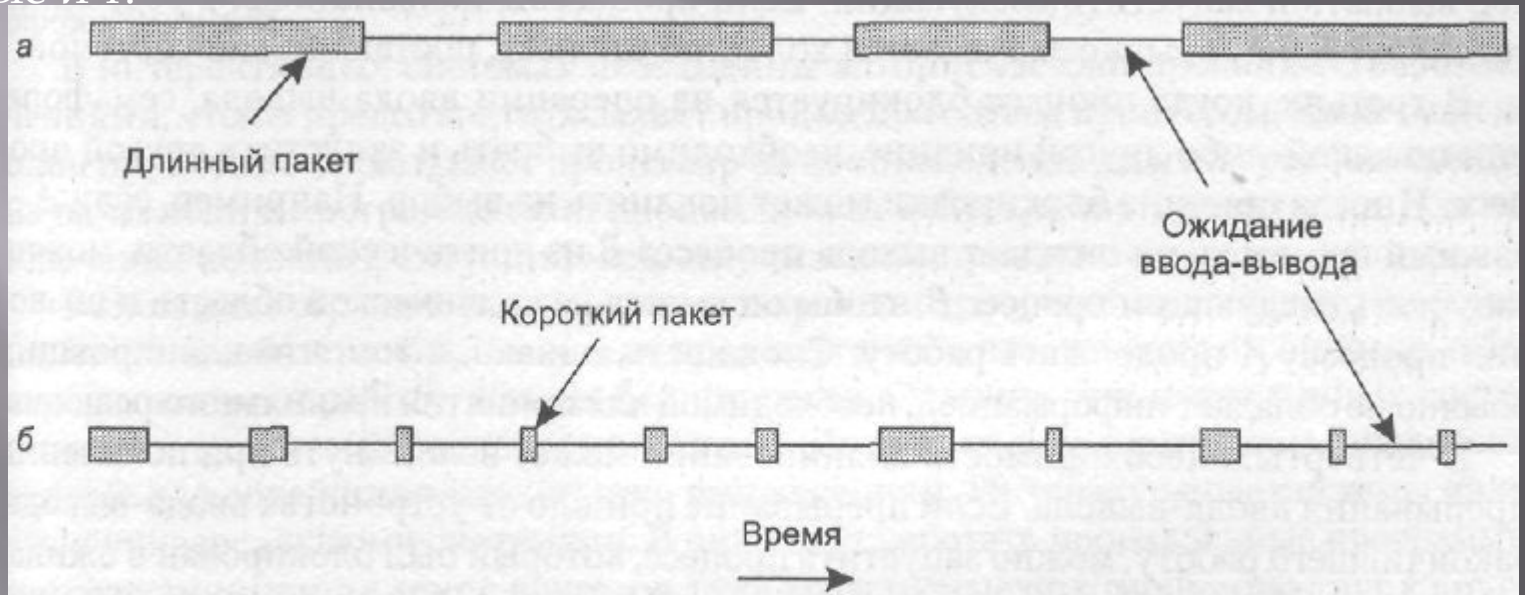


Рис. 2.20. Периоды использования процессора, чередующиеся с ожиданием ввода-вывода: процесс, ограниченный возможностями процессора (*а*); процесс, ограниченный возможностями устройств ввода-вывода (*б*)

Когда планировать?

1. Когда создается новый процесс, необходимо решить, какой процесс запустить: родительский или дочерний.
2. Когда процесс завершает работу. Этот процесс уже не существует, следовательно, необходимо из набора готовых процессов выбрать и запустить следующий
3. Когда процесс блокируется на операции ввода-вывода, семафоре, или по какой-либо другой причине, необходимо выбрать и запустить другой процесс
4. При появлении прерывания ввода-вывода. Планировщик должен выбрать, какой процесс запустить: новый, тот, который был остановлен прерыванием, или какой-то другой.

Категории алгоритмов планирования

Категории алгоритмов планирования согласно их поведению после прерываний.

1. **Алгоритмы планирования без переключений (неприоритетное планирование)**, выбирают процесс и позволяют ему работать вплоть до блокировки (в ожидании ввода-вывода или другого процесса), либо вплоть до того момента, когда процесс сам не отдаст процессор. Процесс не будет прерван, даже если он работает часами.
2. **Алгоритмы планирования с переключениями (приоритетное планирование)**, выбирают процесс и позволяют ему работать некоторое максимально возможное фиксированное время.

Категории алгоритмов планирования, в зависимости от среды

1. **Системы пакетной обработки данных** - нет пользователей, сидящих за терминалами и ожидающих ответа. В таких системах приемлемы алгоритмы без переключений или с переключениями, но с большим временем, отводимым каждому процессу. Такой метод уменьшает количество переключений между процессами и улучшает эффективность.
2. **Интерактивные системы** - В интерактивных системах необходимы алгоритмы планирования с переключениями, чтобы предотвратить захват процессора одним процессом. Даже если ни один процесс не захватывает процессор на неопределенно долгий срок намеренно, из-за ошибки в программе один процесс может заблокировать остальные.
3. **Системы реального времени** - процессы знают, что их время ограничено, и быстро выполняют работу, а затем блокируются. работают только программы, предназначенные для содействия конкретным

Задачи алгоритмов планирования

Для всех систем свойственны:

Справедливость — предоставление каждому процессу справедливой доли процессорного времени

Принудительное применение политики — контроль за выполнением принятой политики

Баланс — поддержка занятости всех частей системы

Системы пакетной обработки данных

Пропускная способность — максимальное количество задач в час

Оборотное время — минимизация времени, затрачиваемого на ожидание обслуживания и обработку задачи

Использование процессора — поддержка постоянной занятости процессора

Интерактивные системы

Время отклика — быстрая реакция на запросы

Соразмерность — выполнение пожеланий пользователя

Системы реального времени

Окончание работы к сроку — предотвращение потери данных

Предсказуемость — предотвращение деградации качества в мультимедийных системах

Планирование в системах пакетной обработки данных

«Первым пришел — первым обслужен»

Процессам предоставляется доступ к процессору в том порядке, в котором они его запрашивают. Чаще всего формируется единая очередь ждущих процессов. Когда текущий процесс блокируется, запускается следующий в очереди, а когда блокировка снимается, процесс попадает в конец очереди.

Достоинства:

1. Легко понять и столь же легко программировать.
2. Все процессы в состоянии готовности контролируются одним связным списком.

Недостаток:

1. Представьте себе, что есть один процесс, ограниченный возможностями процессора, который каждый раз работает ровно 1 с, и много процессов, ограниченных возможностями устройств ввода-вывода, каждый из которых очень в небольшой мере использует процессор, но должен выполнить 1000 обращений к диску.

«Кратчайшая задача — первая»

Временные отрезки работы известны заранее. Если в очереди есть несколько одинаково важных задач, планировщик выбирает первой самую короткую задачу. У нас есть четыре задачи: Л, Л, С и D, со временем выполнения 8, 4, 4 и 4 мин соответственно. Если мы запустим их в данном порядке, оборотное время задачи Л будет 8 мин, В — 12 мин, С — 16 мин и D — 20 мин, и среднее время будет равно 14 мин.

а) запуск четырех задач в исходном порядке б) запуск в соответствии с алгоритмом планирования

Недостаток:
Схема работает лишь в случае одновременного наличия задач.

Наименьшее оставшееся время выполнения

Планировщик каждый раз выбирает процесс с наименьшим оставшимся временем выполнения. Когда поступает новая задача, ее полное время выполнения сравнивается с оставшимся временем выполнения текущей задачи. Если время выполнения новой задачи меньше, текущий процесс приостанавливается и управление передается новой задаче.

Достоинство:

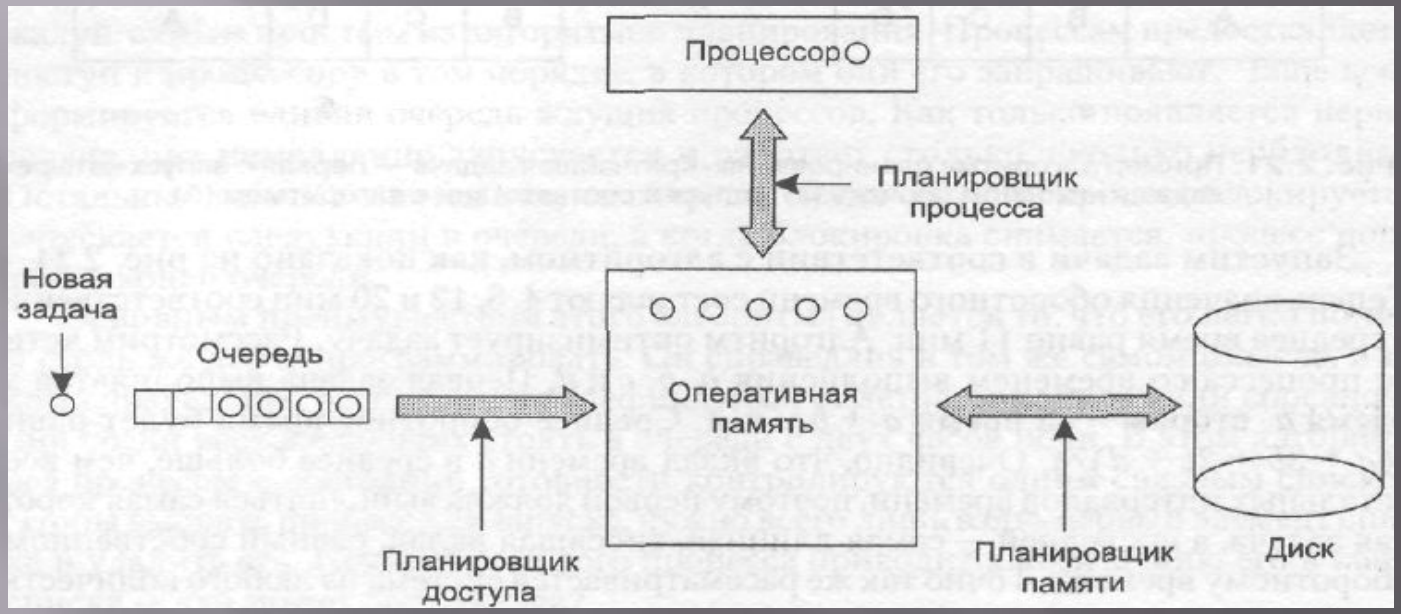
Позволяет быстро обслуживать короткие запросы.

Недостатки:

Необходимо заранее знать время выполнения задач

Трехуровневое планирование

По мере поступления в систему новые задачи сначала помещаются в очередь, хранящуюся на диске. Впускной планировщик выбирает задание и передает его системе. Остальные задания остаются в очереди



Как только задание попало в систему, для него будет создан соответствующий процесс, и он может тут же вступить в борьбу за доступ к процессору. Второй уровень планирования определяет, какие процессы можно хранить в памяти, а какие — на диске. Этим занимается планировщик памяти.

Степень многозадачности - количество процессов, одновременно находящихся в памяти,

Планирование в интерактивных системах

Циклическое планирование

Каждому процессу предоставляется некоторый интервал времени процессора, так называемый квант времени. Если к концу кванта времени процесс все еще работает, он прерывается, а управление передается другому процессу. Планировщику нужно всего лишь поддерживать список процессов в состоянии готовности. Когда процесс исчерпал свой лимит времени, он отправляется в конец списка.

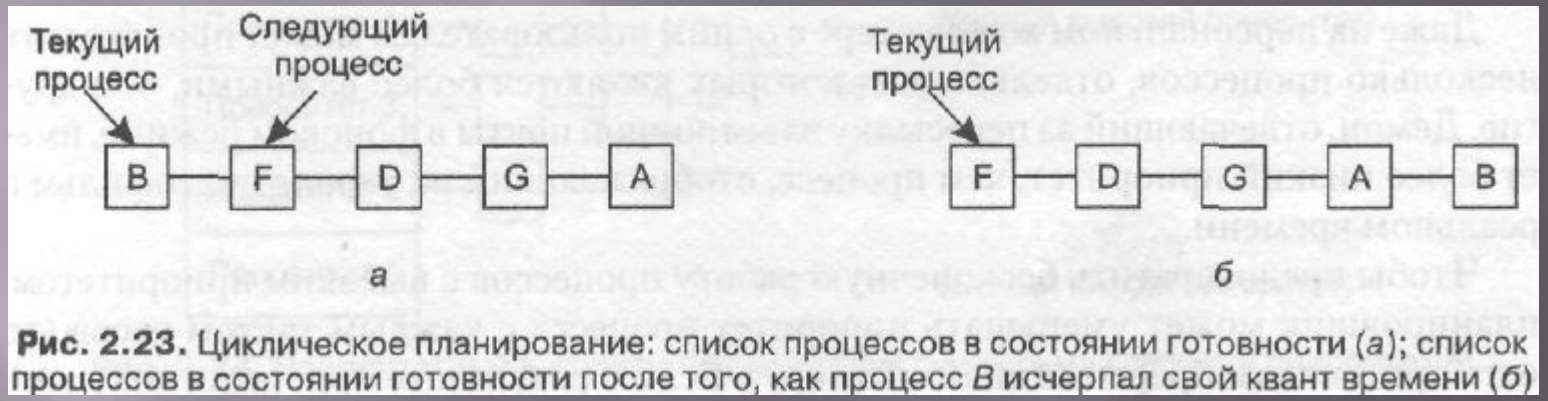


Рис. 2.23. Циклическое планирование: список процессов в состоянии готовности (*а*); список процессов в состоянии готовности после того, как процесс *B* исчерпал свой квант времени (*б*)

Если установленное значение кванта больше среднего интервала работы процессора, переключение процессов будет происходить редко. Напротив, большинство процессов будут совершать блокирующую операцию прежде, чем истечет длительность кванта, вызывая переключение процессов. Устранение принудительных переключений процессов улучшает производительность системы, так как переключения процессов будут происходить только тогда, когда это логически необходимо, то есть когда процесс заблокировался и не может продолжать работу.

Приоритетное планирование

Каждому процессу присваивается приоритет, и управление передается готовому к работе процессу с самым высоким приоритетом.

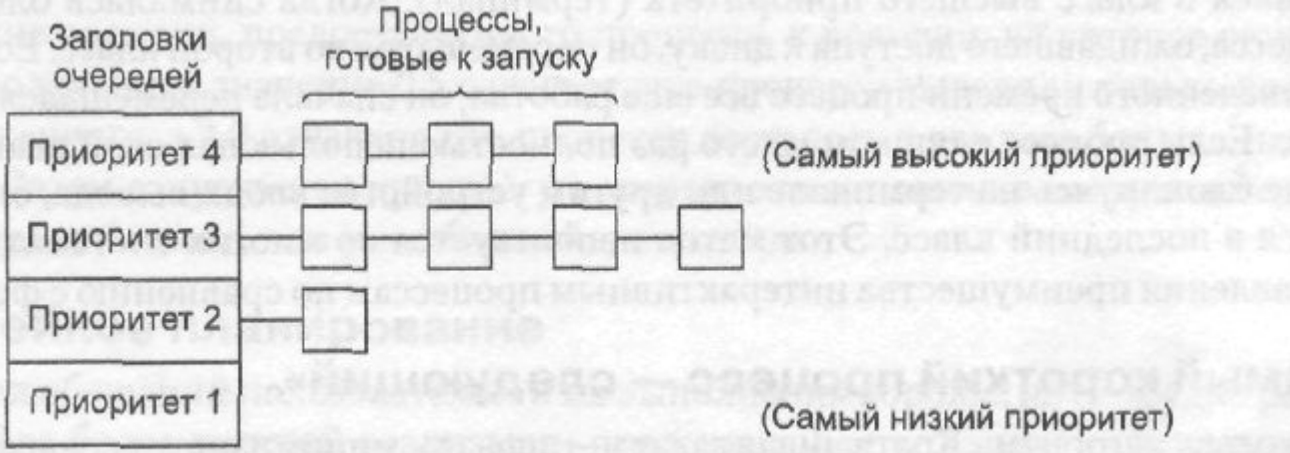


Рис. 2.24. Приоритетный алгоритм планирования с четырьмя классами приоритетов

Чтобы предотвратить бесконечную работу процессов с высоким приоритетом, планировщик может уменьшать приоритет процесса при каждом прерывании

Несколько очередей

Процессам класса с высшим приоритетом выделялся один квант, процессам следующего класса — два кванта, следующего — четыре кванта и т. д. Когда процесс использовал все отведенное ему время, он перемещался на класс ниже.

«Самый короткий процесс — следующий»

Основывается на оценке длины процесса, базирующейся на предыдущем поведении процесса. При этом запускается процесс, у которого оцененное время самое маленькое.

Метод оценки следующего значения серии через взвешенное среднее предыдущего значения и предыдущей оценки часто называют **старением**. Этот метод применим во многих ситуациях, где необходима оценка по предыдущим значениям.

Гарантированное планирование

Система должна отслеживать распределение процессора между процессами с момента создания каждого процесса.

Затем система рассчитывает количество ресурсов процессора, на которое процесс имеет право, например время с момента создания, деленное на п.

Теперь можно сосчитать отношение времени, предоставленного процессу, к времени, на которое он имеет право. Полученное значение 0,5 означает, что процессу выделили только половину положенного, а 2,0 означает, что процессу досталось в два раза больше, чем положено.

Затем запускается процесс, у которого это отношение наименьшее, пока оно не станет больше, чем у его ближайшего соседа.

Лотерейное планирование

В основе алгоритма лежит раздача процессам лотерейных билетов на доступ к различным ресурсам, в том числе и к процессору.

Когда планировщику необходимо принять решение, выбирается случайным образом лотерейный билет, и его обладатель получает доступ к ресурсу.

«Лотерея» может происходить 50 раз в секунду, и победитель получает 20 мс времени процессора.

Достоинство:

1. Обладает высокой отзывчивостью.
2. Взаимодействующие процессы могут при необходимости обмениваться билетами.
3. Позволяет решать задачи, которые не решить с помощью других алгоритмов.
4. Можно реализовать загрузку процессора в желаемой пропорции

Справедливое планирование

В такой модели каждому пользователю достается некоторая доля процессора, и планировщик выбирает процесс в соответствии с этим фактом.

Если в нашем примере каждому из пользователей было обещано по 50 % процессора, то им достанется по 50 % процессора, независимо от количества процессов.

Планирование в системах реального времени

Системы реального времени делятся:

1. **На жесткие системы реального времени**, что означает наличие жестких сроков для каждой задачи (в них обязательно надо укладываться),
2. **На гибкие системы реального времени**, в которых нарушения временного графика нежелательны, но допустимы.

Внешние события, на которые система должна реагировать, можно разделить на:

1. П**ериодические** (возникающие через регулярные интервалы времени)
2. **Непериодические** (возникающие непредсказуемо).

Алгоритмы планирования для систем реального времени могут быть

1. Статическими.

Решения планирования принимаются заранее, еще до запуска системы.

Статическое планирование применимо только при наличии достоверной информации о работе, которую необходимо выполнить, и о временном графике, которого нужно придерживаться.

2. Динамическими.

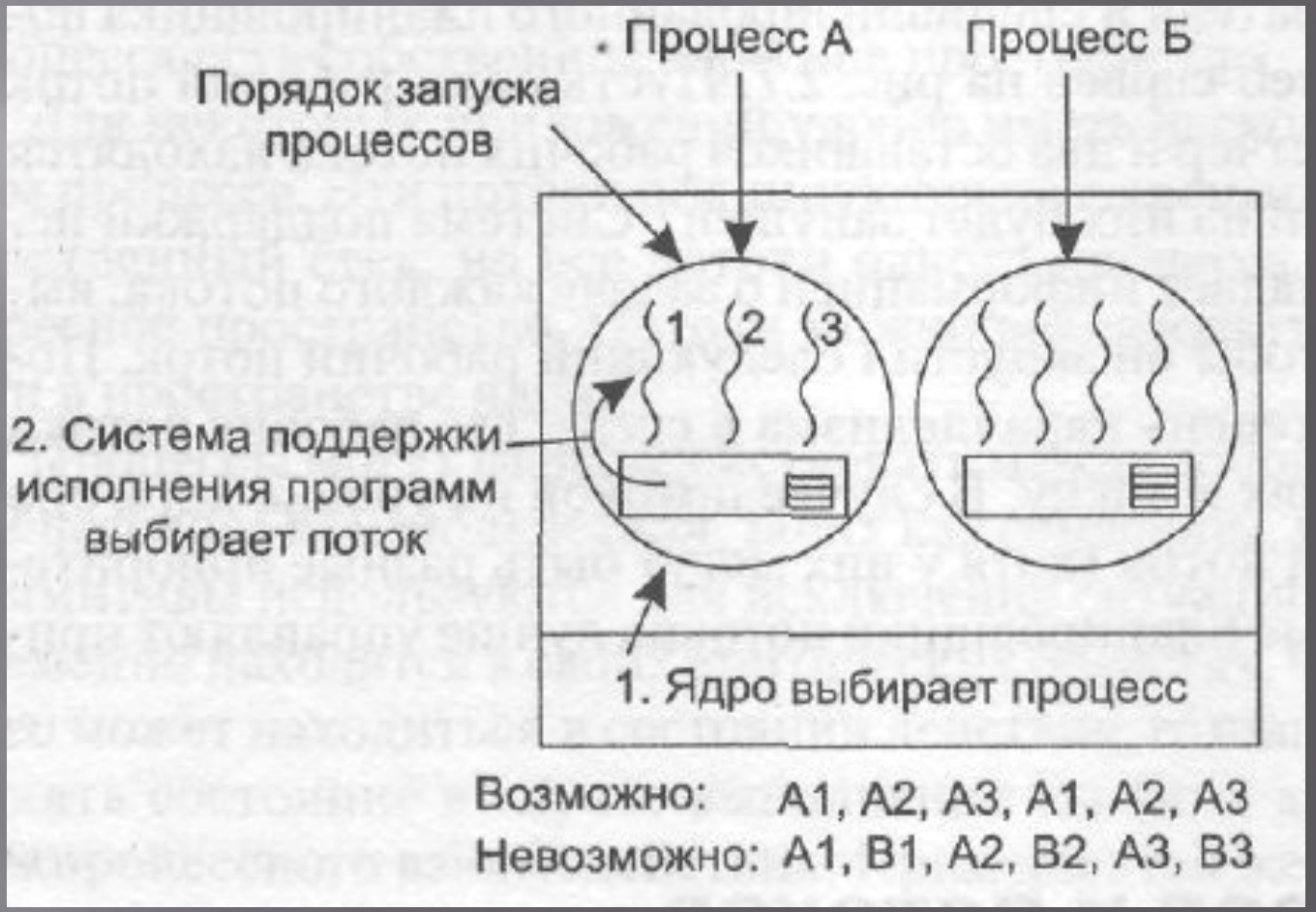
Решения планирования принимаются по ходу дела.

Динамическое планирование не нуждается в

Планирование потоков

Потоки на уровне пользователя.

Поскольку ядро не знает о существовании потоков, оно выполняет обычное планирование, выбирая процесс Л и предоставляя ему квант времени. Планировщик потоков внутри процесса Л выбирает поток, например А1.



В качестве алгоритма планирования для системы поддержки исполнения программ можно взять любой из уже рассмотренных нами.

Потоки на уровне ядра

ядро выбирает следующий поток. При этом ядро не обязано принимать во внимание, какой поток принадлежит какому процессу, хотя у него есть такая возможность. Потоку предоставляется квант времени и по истечении этого кванта управление передается другому потоку.