Типы планирования. Алгоритмы планирования. Примеры реализации алгоритмов планирования в современных операционных системах.

Планирование - обеспечение поочередного доступа процессов к одному процессору.

Планировщик - отвечающая за это часть операционной системы.

Алгоритм планирования - используемый алгоритм для планирования.

Ситуации, когда необходимо планирование:

- Когда создается процесс
- Когда процесс завершает работу
- Когда процесс блокируется на операции ввода/вывода, семафоре, и т.д.
- При прерывании ввода/вывода.

Алгоритм планирования без переключений (неприоритетный) - не требует прерывание по аппаратному таймеру, процесс останавливается только когда блокируется или завершает работу.

Алгоритм планирования с переключениями (приоритетный) - требует прерывание по аппаратному таймеру, процесс работает только отведенный период времени, после этого он приостанавливается по таймеру, чтобы передать управление планировщику.

Необходимость алгоритма планирования зависит от задач, для которых будет использоваться операционная система.

В многозадачных системах в основной памяти одновременно содержится код нескольких процессов. В работе каждого процесса периоды использования процессора чередуются с ожиданием завершения выполнения операций ввода-вывода или некоторых внешних событий. Процессор (или процессоры) занят выполнением одного процесса, в то время как остальные находятся в состоянии ожидания.

Ключом к многозадачности является планирование. Обычно используются четыре типа планирования:

Долгосрочное	**Решение о добавлении процесса в пул выполняемых процессов**
планирование	
Среднесрочное	Решение о добавлении процесса к числу процессов, полностью или частично
планирование	размещённых в памяти
Краткосрочное	Решение о том, какой из доступных процессов будет выполняться процессором
планирование	
Планирование	Решение о том, какой из запросов процессоров на операции ввода-вывода
ввода-вывода	будет обработан свободным устройством
	ввода-вывода

Задачи алгоритмов планирования:

Для всех систем

Справедливость - каждому процессу справедливую долю процессорного времени

Контроль над выполнением принятой политики
Баланс - поддержка занятости всех частей системы (например: чтобы были заняты процессор и устройства ввода/вывода)

Системы пакетной обработки

Пропускная способность - количество задач в час

Оборотное время - минимизация времени на ожидание обслуживания и обработку задач.

Использование процесса - чтобы процессор всегда был занят.

Интерактивные системы

Время отклика - быстрая реакция на запросы Соразмерность - выполнение ожиданий пользователя (например: пользователь не готов к долгой загрузке системы)

Системы реального времени

Окончание работы к сроку - предотвращение потери данных

Предсказуемость - предотвращение деградации качества в мультимедийных системах (например: потерь качества звука должно быть меньше чем видео)

Планирование в системах пакетной обработки

"Первый пришел - первым обслужен" (FIFO - First In Fist Out)

Процессы ставятся в очередь по мере поступления.

Преимущества:

* Простата
* Справедливость (как в очереди покупателей, кто последний пришел, тот оказался в конце очереди)

Недостатки:

* Процесс, ограниченный возможностями процессора может затормозить более быстрые процессы, ограниченные устройствами ввода/вывода.

Кратчайшая задача - первая«

Нижняя очередь выстроена с учетом этого алгоритма

Преимущества:

-Уменьшение оборотного времени

-Справедливость (как в очереди покупателей, кто без сдачи проходит в перед)

Недостатки:

- Длинный процесс, занявший процессор, не пустит более новые краткие процессы, которые пришли позже.

Наименьшее оставшееся время выполнения

Аналог предыдущего, но если приходит новый процесс, его полное время выполнения сравнивается с оставшимся временем выполнения текущего процесса.

Трехуровневое планирование

Планировщик доступа выбирает задачи оптимальным образом(например: процессы, ограниченные процессором и вводом/выводом).

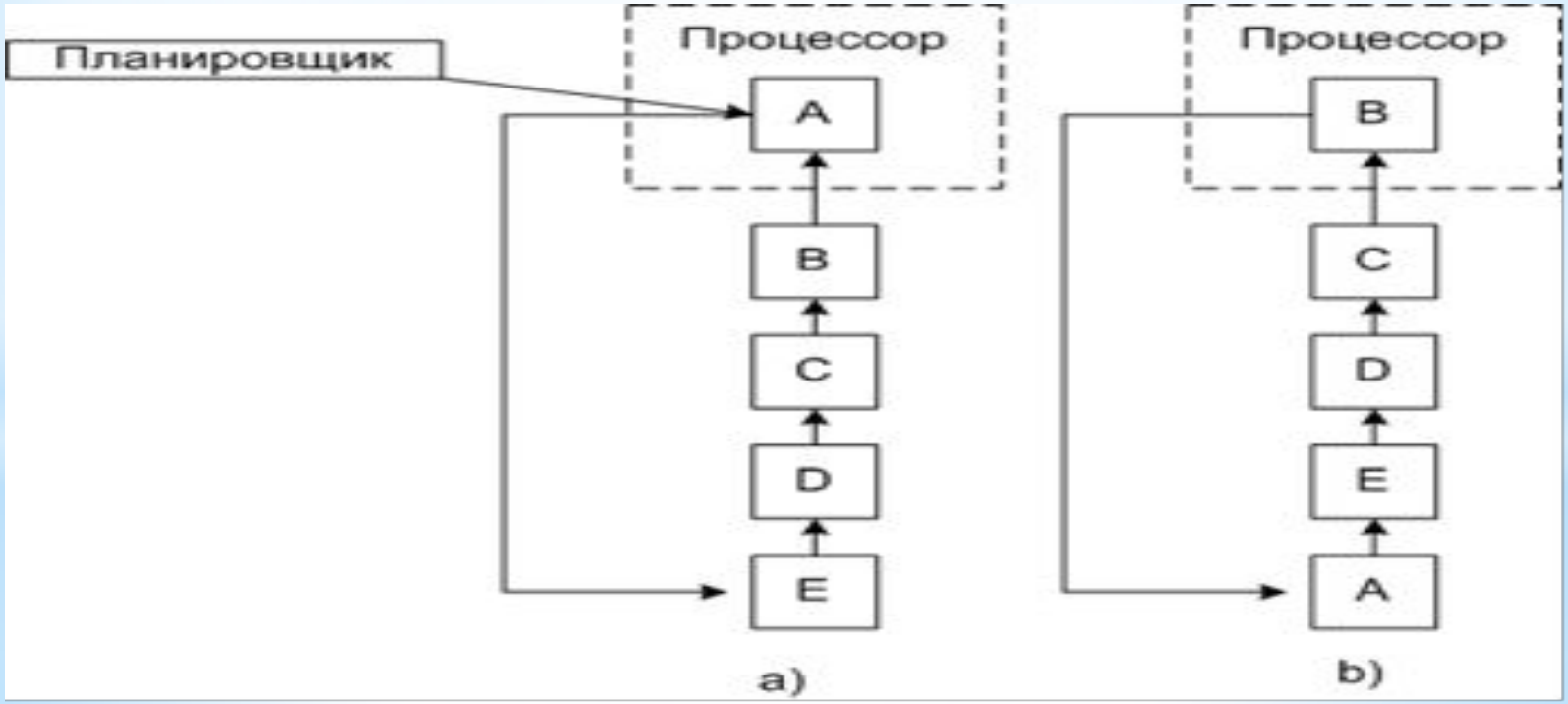
Если процессов в памяти слишком много, планировщик памяти выгружает и загружает некоторые процессы на диск. Количество процессов находящихся в памяти, называется **степенью многозадачности**.

Планирование в интерактивных системах

Циклическое планирование

Самый простой алгоритм планирования и часто используемый.

Каждому процессу предоставляется квант времени процессора. Когда квант заканчивается процесс переводится планировщиком в конец очереди. При блокировке процесс выпадает из очереди.



Преимущества:

* Простата
* Справедливость (как в очереди покупателей, каждому только по килограмму)

Недостатки:

* Если частые переключения (квант - 4мс, а время переключения равно 1мс), то происходит уменьшение производительности.
* Если редкие переключения (квант - 100мс, а время переключения равно 1мс), то происходит увеличение времени ответа на запрос.

Приоритетное планирование

Каждому процессу присваивается **приоритет**, и управление передается процессу с самым высоким приоритетом.

Приоритет может быть **динамический и статический**.

Динамический приоритет может устанавливаться так:

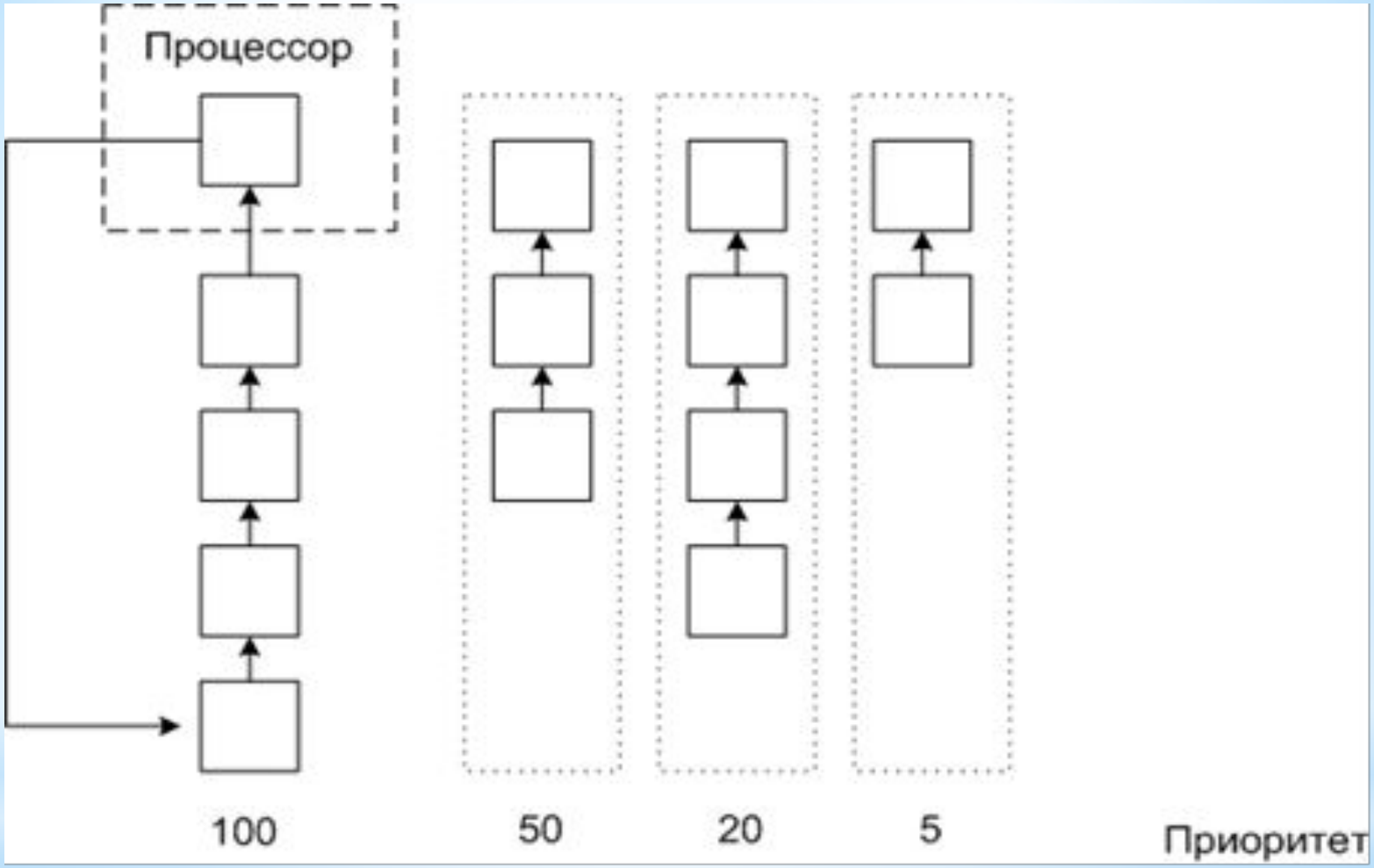
П=1/Т, где Т- часть использованного в последний раз кванта

Если использовано 1/50 кванта, то приоритет 50.

Если использован весь квант, то приоритет 1.

Т.е. процессы, ограниченные вводом/вывода, будут иметь приоритет над процессами ограниченными процессором.

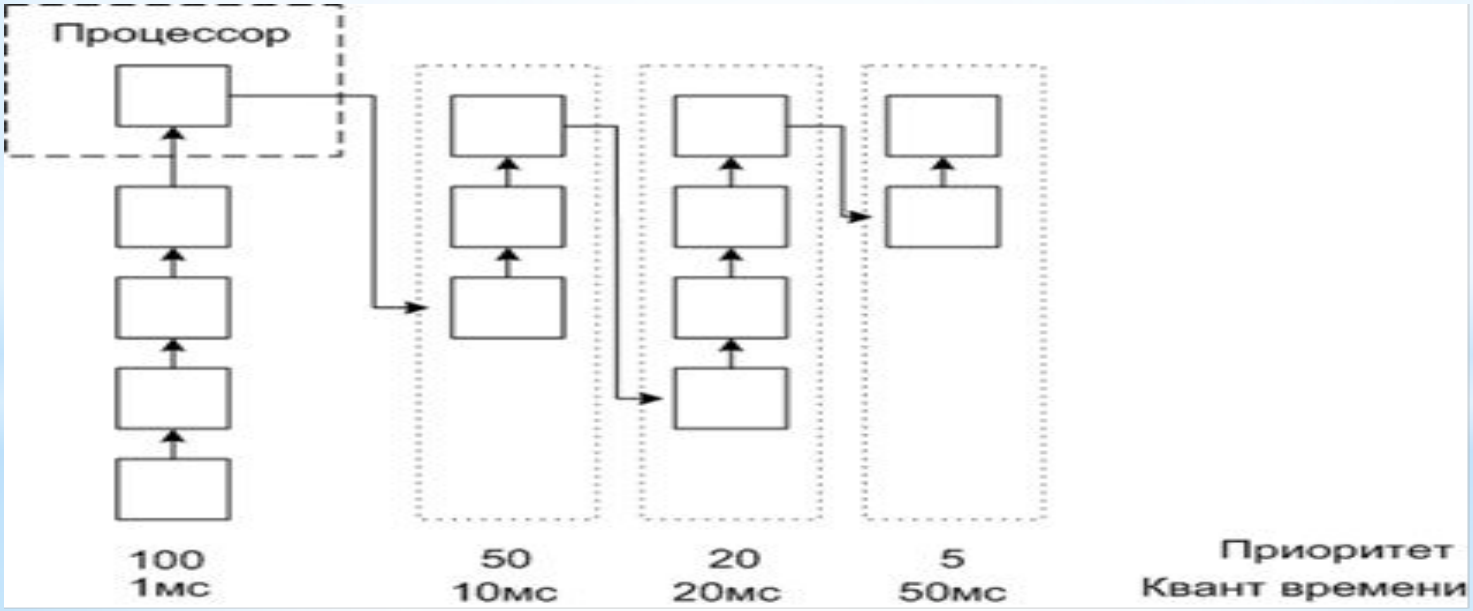
Часто процессы объединяют по приоритетам в группы, и используют приоритетное планирование среди групп, но внутри группы используют циклическое планирование.


Процессор
100
50
20
5
Приоритет

Методы разделения процессов *на группы*

Группы с разным квантом времени

Сначала процесс попадает в группу с наибольшим приоритетом и наименьшим квантом времени, если он использует весь квант, то попадает во вторую группу и т.д. Самые длинные процессы оказываются в группе наименьшего приоритета и наибольшего кванта времени



Планирование в системах реального времени

Системы реального времени делятся на:

- ***жесткие*** (жесткие сроки для каждой задачи) - управление движением
- ***гибкие*** (нарушение временного графика не желательны, но допустимы) - управление видео и аудио

Внешние события, на которые система должна реагировать, делятся:

- ***периодические*** - потоковое видео и аудио
- ***непериодические*** (непредсказуемые) - сигнал о пожаре

Планирование однородных процессов

В качестве однородных процессов можно рассмотреть **видео сервер с несколькими видео потоками** (несколько пользователей смотрят фильм).

Т.к. все процессы важны, можно использовать циклическое планирование.

Но так как количество пользователей и размеры кадров могут меняться, для реальных систем он не подходит.

Общее планирование реального времени

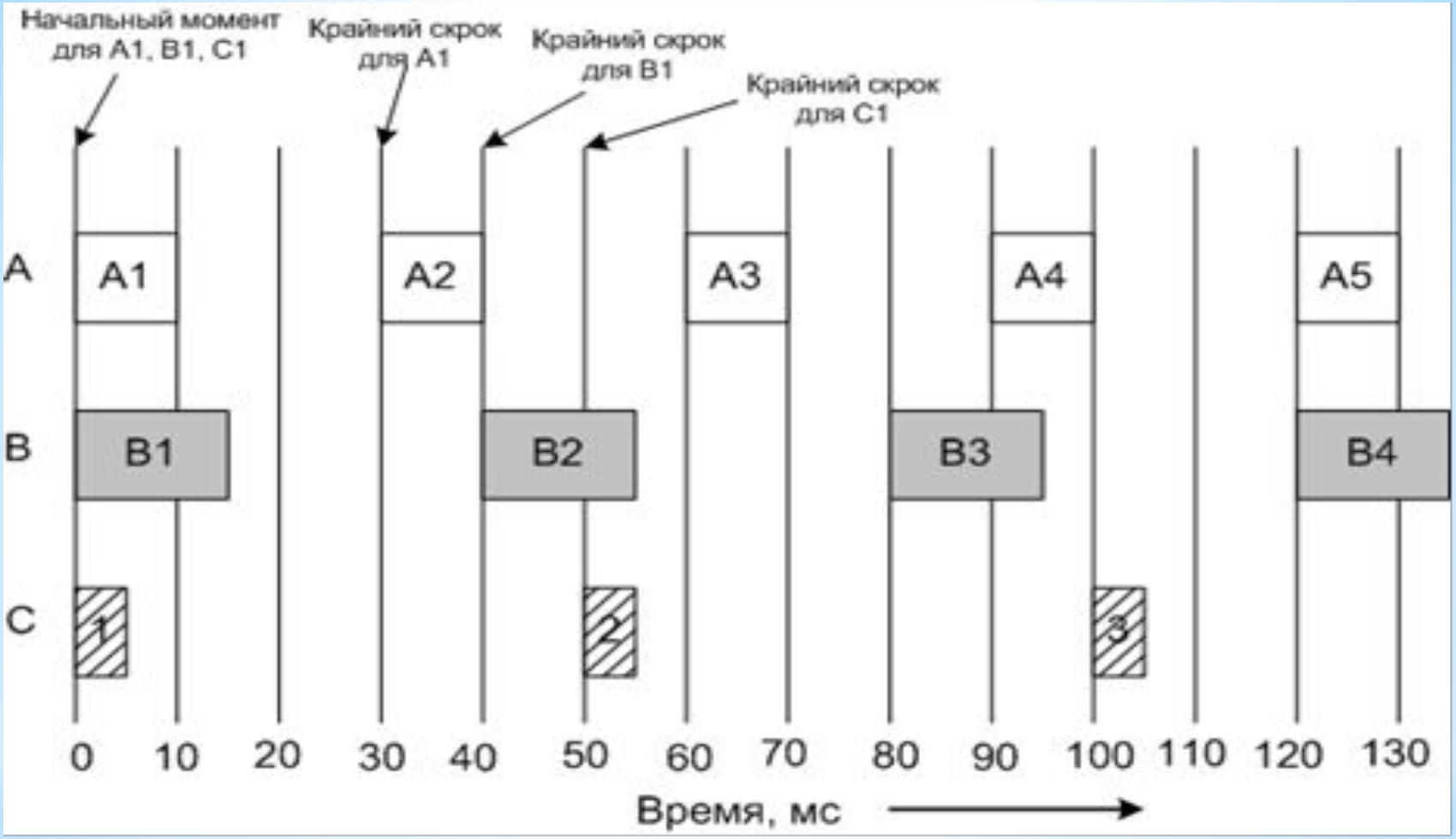
Используется модель, когда каждый процесс борется за процессор со своим заданием и графиком его выполнения.

Планировщик должен знать:

- частоту, с которой должен работать каждый процесс
- объем работ, который ему предстоит выполнить
- ближайший срок выполнения очередной порции задания

Рассмотрим ***пример из трех процессов***.

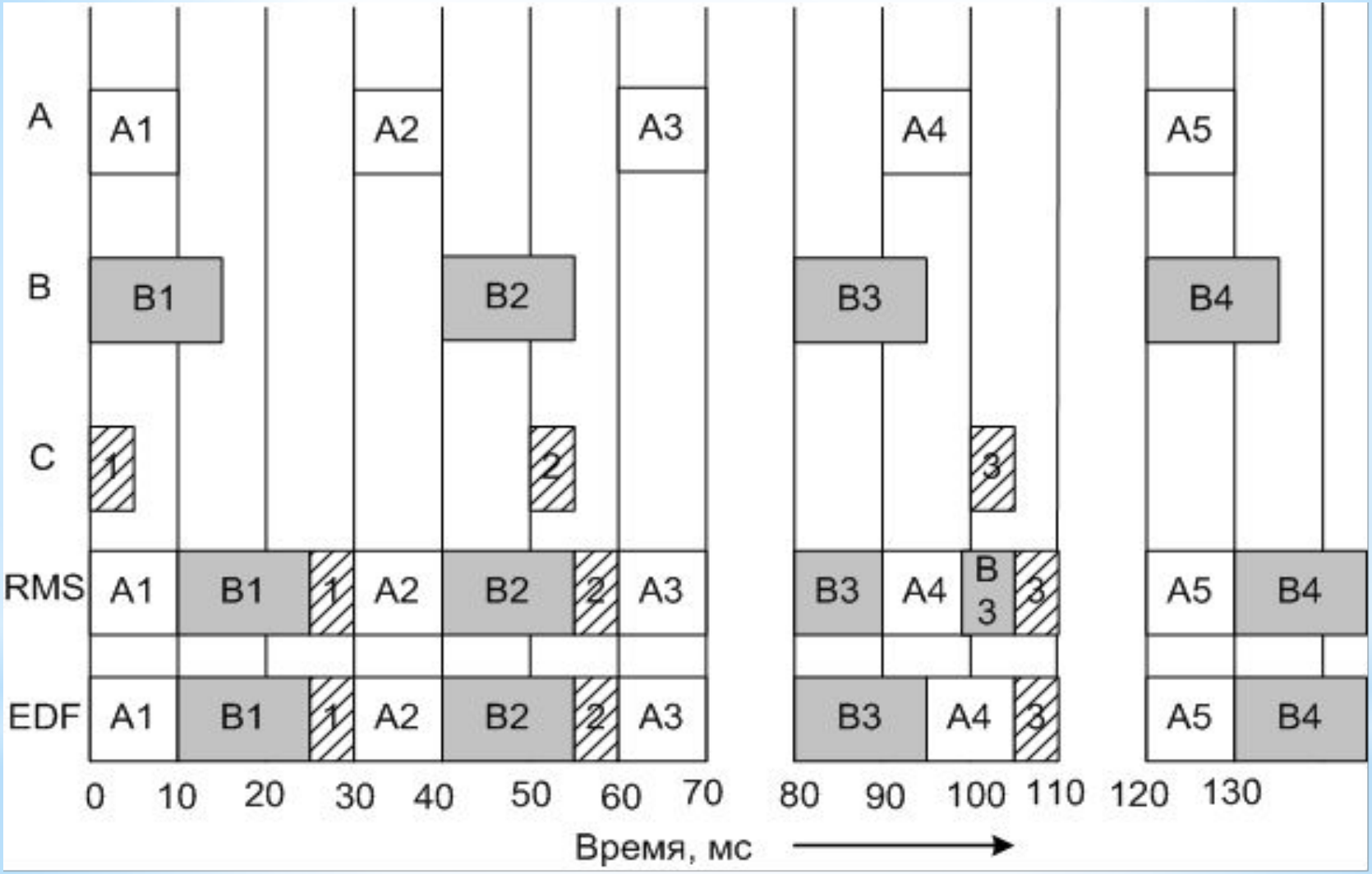
* Процесс **A** запускается каждые 30мс, обработка кадра 10мс
* Процесс **B** частота 25 кадров, т.е. каждые 40мс, обработка кадра 15мс
* Процесс **C** частота 20 кадров, т.е. каждые 50мс, обработка кадра 5мс


Начальный момент для A1, B1, C1
Крайний скрок для A1
Крайний скрок для B1
Крайний скрок для C1
A
B
C
A1
A2
A3
A4
A5
B1
B2
B3
B4
1
2
3
0
10
20
30
40
50
60
70
80
90
100
110
120
130
Время, мс

Статический алгоритм планирования RMS (Rate Monotonic Scheduling)

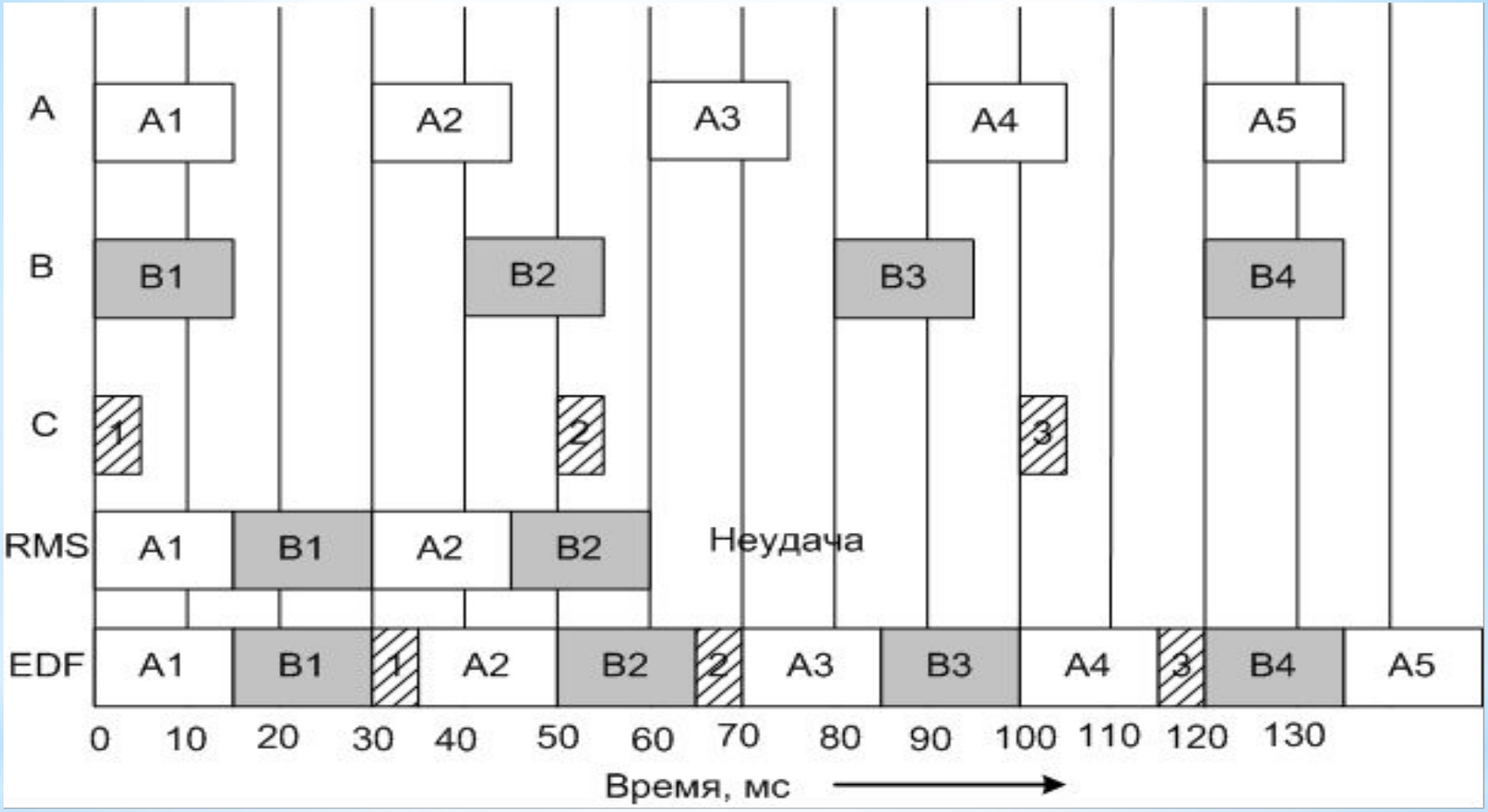
Процессы должны ***удовлетворять условиям***:
- Процесс должен быть завершен за время его периода
- Один процесс не должен зависеть от другого
- Каждому процессу требуется одинаковое процессорное время на каждом интервале
- У непериодических процессов нет жестких сроков
- Прерывание процесса происходит мгновенно

Приоритет в этом алгоритме пропорционален частоте.
Процессу А он равен 33 (частота кадров)
Процессу В он равен 25
Процессу С он равен 20
Процессы выполняются по приоритету


A
B
C
RMS
EDF
A1
A2
A3
A4
A5
B1
B2
B3
B4
1
2
3
B 3
0 10 20 30 40 50 60 70 80 90 100 110 120 130
Время, мс

Динамический алгоритм планирования EDF (Earliest Deadline First)

Наибольший приоритет выставляется процессу, у которого осталось наименьшее время выполнения.
При больших загрузках системы **EDF** имеет преимущества.
Рассмотрим пример, когда процессу А требуется для обработки кадра - 15мс.



Алгоритм планирования **RMS** терпит неудачу.