4.1 Основные понятия планирования процессов

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:

1. Когда создается процесс
2. Когда процесс завершает работу
3. Когда процесс блокируется на операции ввода/вывода, семафоре, и т.д.
4. При прерывании ввода/вывода.

Алгоритм планирования без переключений (неприоритетный) - не требует прерывание по аппарат. 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.

Основные три системы:

1. Системы пакетной обработки - могут использовать неприоритетный и приоритетный алгоритм (например: для расчетных программ).
2. Интерактивные системы - могут использовать только приоритетный алгоритм, нельзя допустить чтобы один процесс занял надолго процессор (например: сервер общего доступа или персональный компьютер).
3. Системы реального времени - могут использовать неприоритетный и приоритетный алгоритм (например: система управления автомобилем).

Задачи алгоритмов планирования:

1. **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)
2. **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.
3. **Интерактивные системы**
   Время отклика - быстрая реакция на запросы
   Соразмерность - выполнение ожиданий пользователя (например: пользователь не готов к долгой загрузке системы)
4. **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)

4.2 Планирование в системах пакетной обработки

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

Процессы ставятся в очередь по мере поступления.

Преимущества:

- Простота
- 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.

4.2.2 "Кратчайшая задача - первая"

4 мин.	6 мин.	2	4 мин.	2	2

2	2	2	4 мин.	4 мин.	6 мин.

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.

4.2.3 Наименьшее оставшееся время выполнение

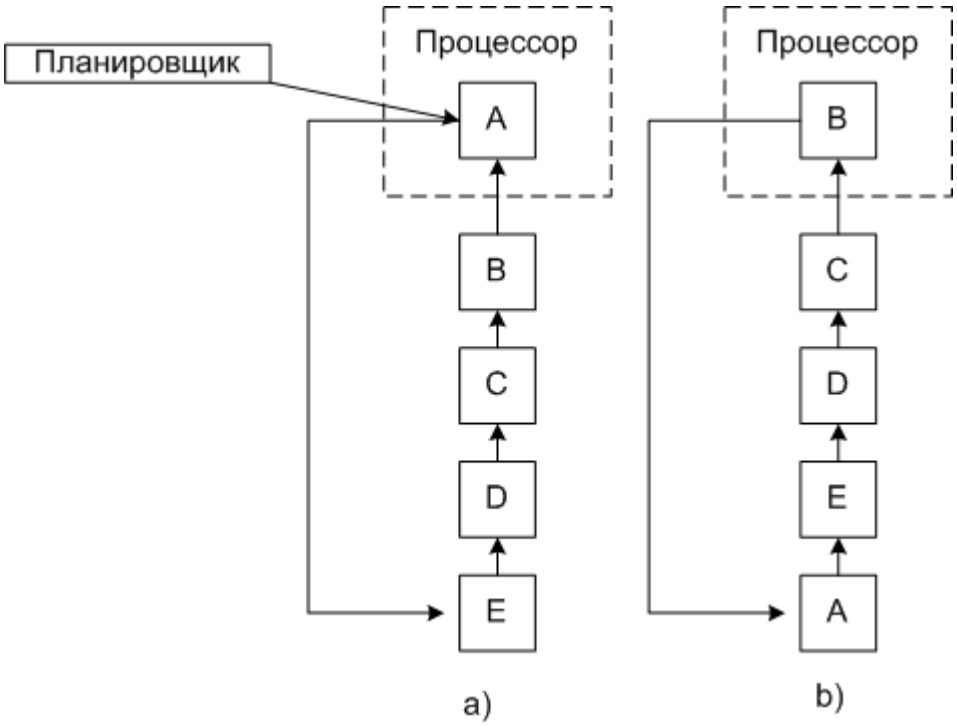
Аналог предыдущего, но если приходит новый процесс, его полное время выполнения сравнивается с оставшимся временем выполнения текущего процесса.

4.3 Планирование в интерактивных системах

4.3.1 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 процессор выпадает из очереди.



Пример циклического планирования

Преимущества:

- Простота
- 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.

4.3.2 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:

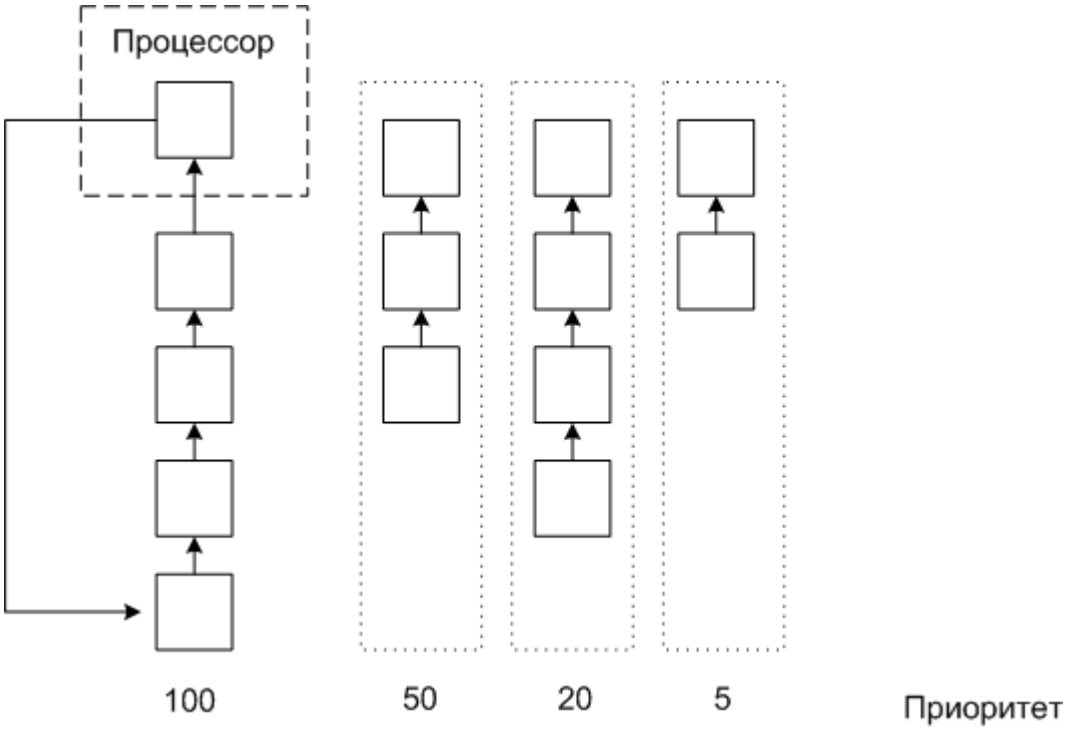
П=1/Т, где Т- часть использованного в последний раз кванта

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.

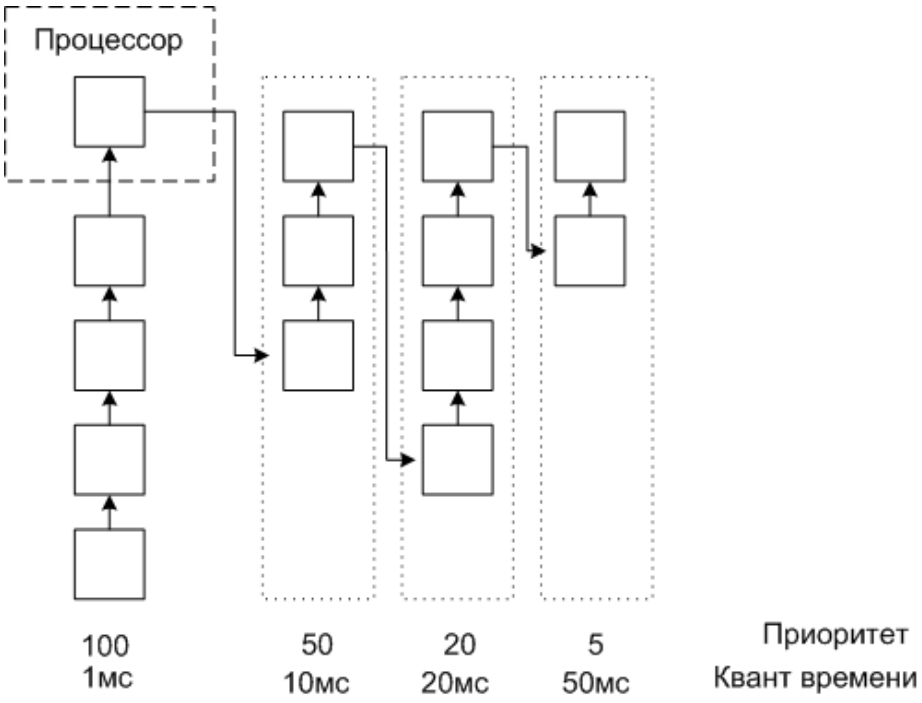


Приоритетное планирование 4-х групп

4.3.3 Методы разделения процессов на группы

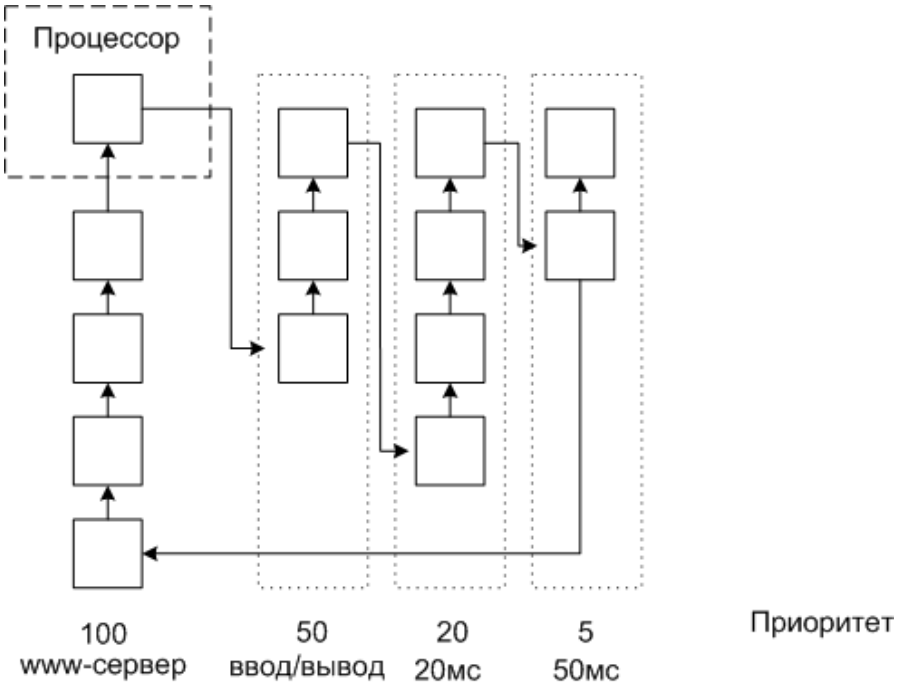
Группы с разным квантом времени

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.



Процесс либо заканчивает работу, либо переходит в другую группу. Этот метод напоминает алгоритм - "Кратчайшая задача - первая".

Группы с разным назначением процессов



Процесс, отвечающий на запрос, переходит в группу с наивысшим приоритетом. Такой механизм позволяет повысить приоритет работы с клиентом.

Гарантированное планирование

В системе с n-процессами, каждому процессу будет предоставлено 1/n времени процессора.

Лотерейное планирование

Процессам раздаются "лотерейные билеты" на доступ к ресурсам. Планировщик может выбрать любой билет, случайным образом. Чем больше билетов у процесса, тем больше у него шансов захватить ресурс.

Справедливое планирование

Процессорное время распределяется среди пользователей, а не процессов. Это справедливо если у одного пользователя несколько процессов, а у другого один.

4.4 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

Что бы систему реального времени можно было планировать, нужно чтобы выполнялось условие:

$$\sum_{i=1}^{m} \frac{T(i)}{P(i)} \leq 1$$

m - число периодических событий

i - номер события

P(i) - период поступления события

T(i) - время, которое уходит на обработку события

Т.е. перегруженная система реального времени является не **планируемой**.

4.4.1 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.

4.4.2 Общее планирование реального времени

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

Планировщик должен знать:

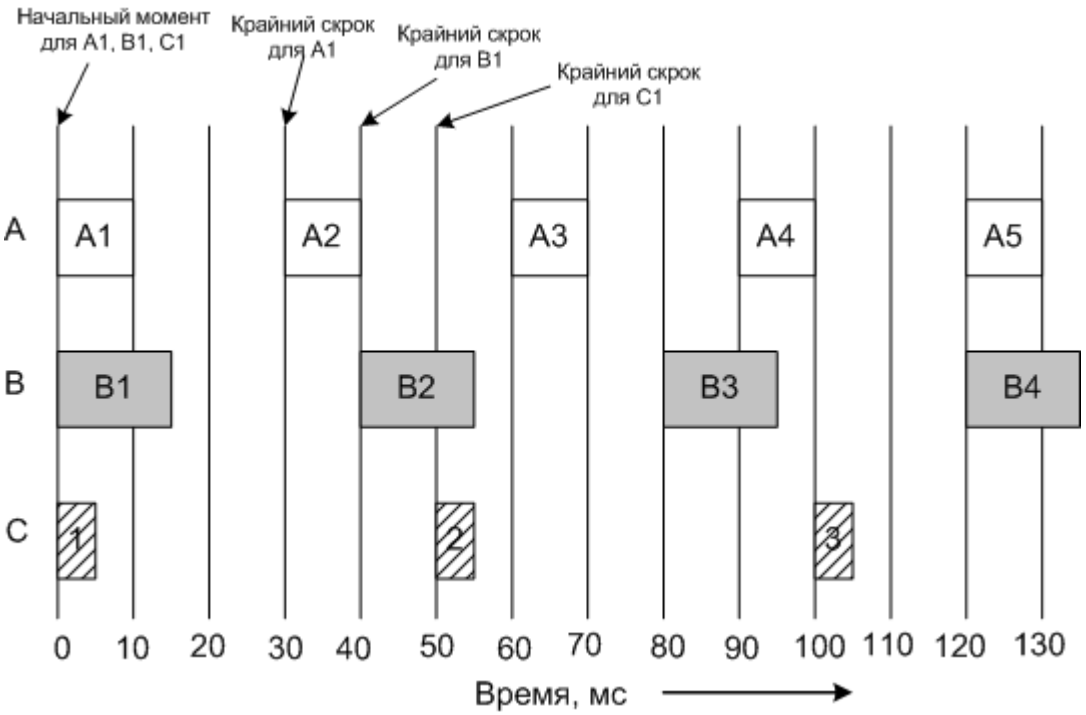
- частоту, с которой должен работать каждый процесс
- объем работ, который ему предстоит выполнить
- ближайший срок выполнения очередной порции задания

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



Три периодических процесса

Проверяем, можно ли планировать эти процессы.

10/30+15/40+5/50=0.808<1

Условие выполняется, планировать можно.

Будем планировать эти процессы **статическим** (приоритет заранее назначается каждому процессу) и **динамическим** методами.

4.4.3 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

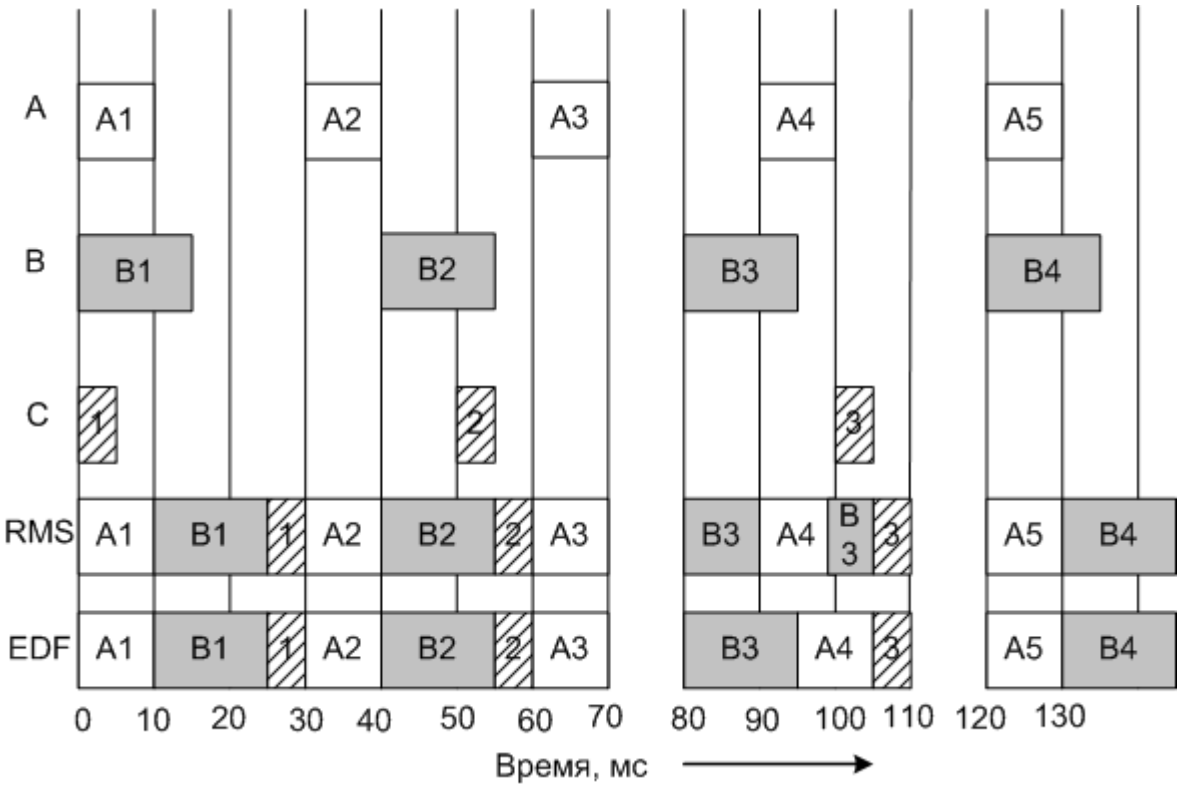
Приоритет в этом алгоритме пропорционален частоте.

Процессу A он равен 33 (частота кадров)

Процессу B он равен 25

Процессу C он равен 20

Процессы выполняются по приоритету.



Статический алгоритм планирования RMS (Rate Monotonic Scheduling)

4.4.4 Динамический алгоритм планирования EDF (Earliest Deadline First)

Наибольший приоритет выставляется процессу, у которого осталось наименьшее время выполнения.

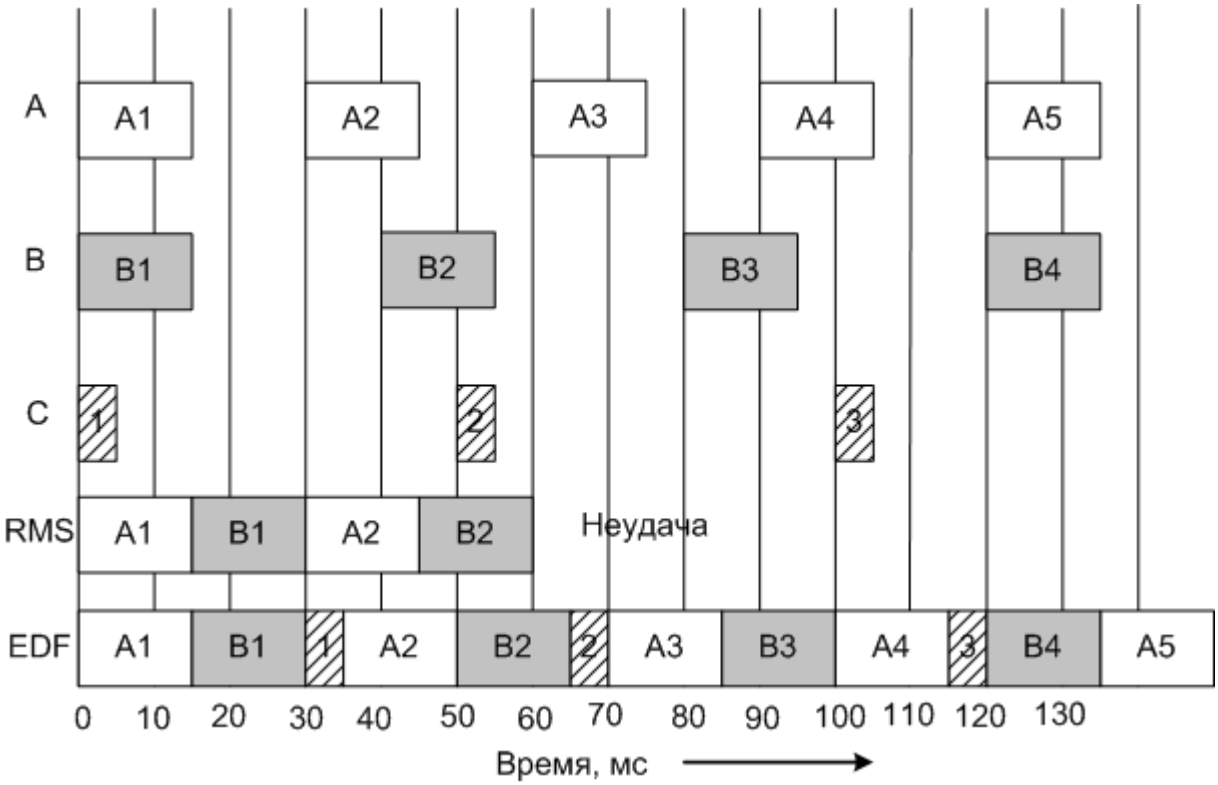
При больших загрузках системы **EDF** имеет преимущества.

Рассмотрим пример, когда процессу A требуется для обработки кадра - 15мс.

Проверяем, можно ли планировать эти процессы.

15/30+15/40+5/50=0.975<1

Загрузка системы 97.5%



Динамический алгоритм планирования EDF (Earliest Deadline First)

Алгоритм планирования **RMS** терпит неудачу.