



# Операционные системы

---

Процессы и потоки. Взаимоблокировки.  
Планирование.

# Взаимоблокировки

---

1. ЧТО?
2. ГДЕ?
3. КОГДА?

# Ресурсы

---

1. Выгружаемые
2. Невыгружаемые

# ХОРОШИЙ КОД

---

## ПРОГРАММА 1

Lock(resource1)

Lock(resource2)

<ACTIONS1...>

UNLOCK(resource1)

UNLOCK(resource2)

## ПРОГРАММА 2

Lock(resource1)

Lock(resource2)

<ACTIONS2...>

UNLOCK(resource1)

UNLOCK(resource2)



# ОПАСНЫЙ КОД

---



## ПРОГРАММА 1

Lock(resource1)

Lock(resource2)

<ACTIONS1...>

UNLOCK(resource1)

UNLOCK(resource2)

## ПРОГРАММА 2

Lock(resource2)

Lock(resource1)

<ACTIONS2...>

UNLOCK(resource1)

UNLOCK(resource2)

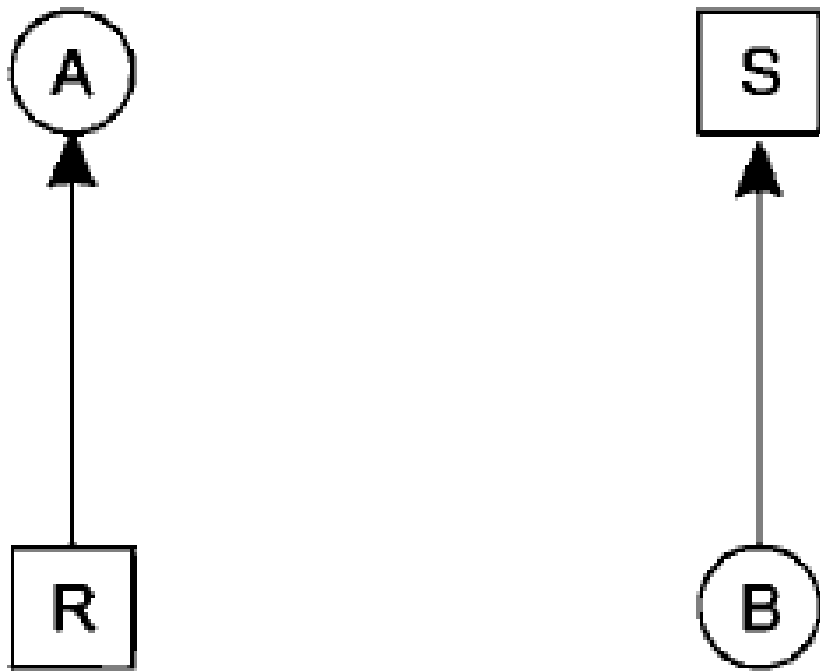
# Условия возникновения взаимоблокировок

---

1. Условие взаимного исключения
2. Условие удержания и ожидания
3. Условие невыгружаемости
4. Условие циклического ожидания

# Моделирование взаимоблокировок

---



## 2 процесса (нет взаимоблокировки)

---

Процесс P1

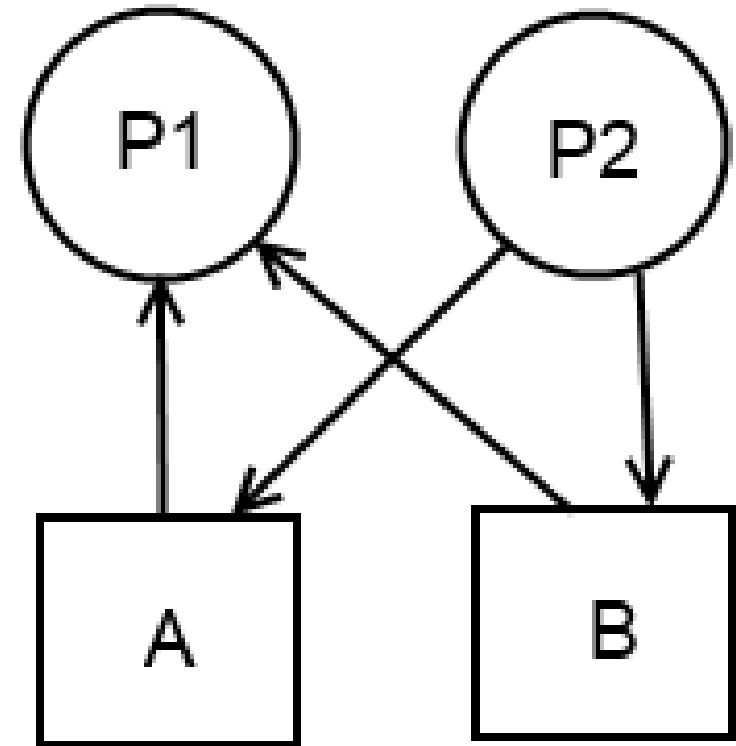
а) Запрошен ресурс A

б) Запрошен ресурс B

Процесс P2

а) Запрошен ресурс A

б) Запрошен ресурс B





## 2 процесса (взаимоблокировка!)

---

Процесс P1

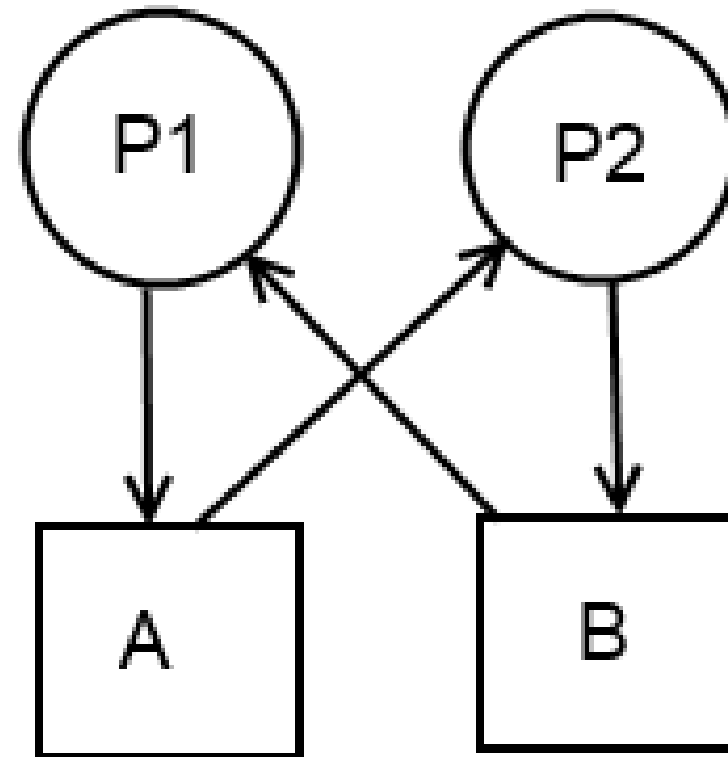
а) Запрошен ресурс B

б) Запрошен ресурс A

Процесс P2

а) Запрошен ресурс A

б) Запрошен ресурс B



# Способы борьбы со взаимоблокировками

---

1. Игнорирование
2. Обнаружение и восстановление
3. Динамическое уклонение
4. Предотвращение за счет подавления условий взаимоблокировок

# Игнорирование

---

1. Самое простое
2. Может быть принято из статистических наблюдений
3. Не «математическое решение»

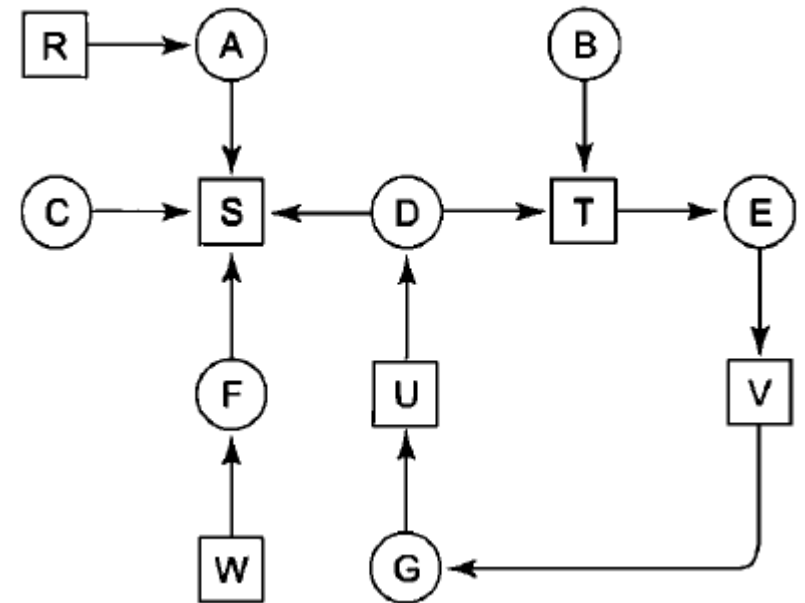
# Обнаружить и восстановить!

---

1. Обнаружение
2. Восстановление работоспособности

# Обнаружение при использовании одного типа ресурса

1. Выбираем произвольную вершину графа
2. Производим обход в ширину
3. Если при обходе встретилась какая-то вершина дважды, то цикл есть



# Обнаружение при использовании ресурсов разных типов

---

A – вектор доступных ресурсов

E – вектор существующих ресурсов

C – матрица текущего распределения

R – матрица запросов

m – количество ресурсов

n – количество процессов

$$\sum_{i=1}^n C_{ij} + A_j = E_j$$

	Накопители на магнитной ленте	Плоттеры	Сканеры	Компакт-диски
$E =$	(4	2	3	1)

	Накопители на магнитной ленте	Плоттеры	Сканеры	Компакт-диски
$A =$	(2	1	0	0)

Матрица текущего  
распределения

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Матрица запросов

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

# Восстановление

---

1. За счет приоритетного овладения ресурса
2. Rollback
3. Уничтожение процесса

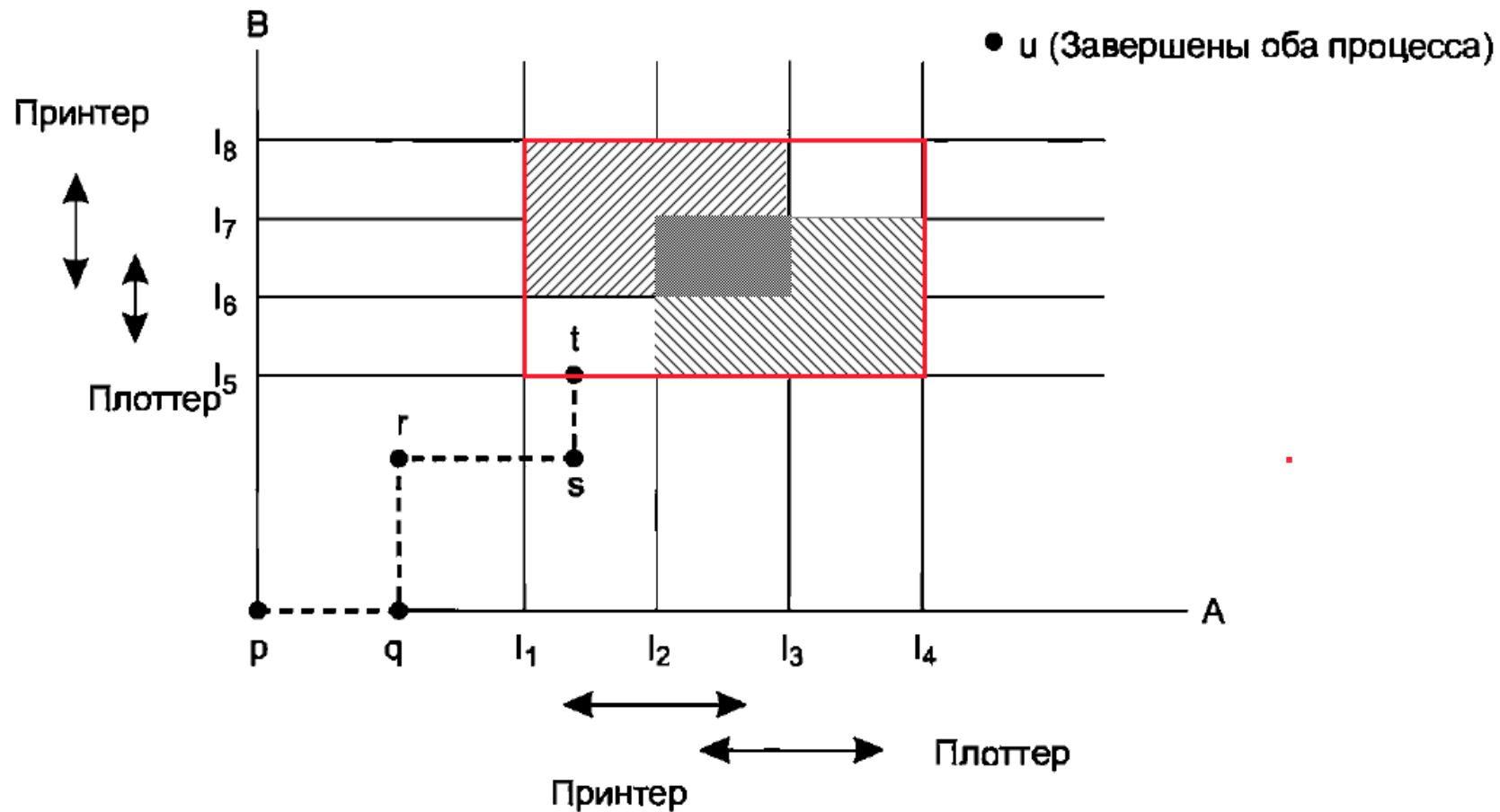


# Уклонение от взаимоблокировок

---

1. Траектория ресурса
2. Поддержание безопасного состояния

# Траектория ресурса



# Безопасное и небезопасное состояние (Алгоритм банкира)

	Имеет	Max
A	0	6
B	0	5
C	0	4
D	0	7

Свободно: 10

	Имеет	Max
A	1	6
B	1	5
C	2	4
D	4	7

Свободно: 2

	Имеет	Max
A	1	6
B	2	5
C	2	4
D	4	7

Свободно: 1

# Алгоритм банкира (несколько типов ресурсов)

Процесс  
Накопители  
на магнитных дисках  
Плоттеры  
Сканеры  
Устройства для чтения  
компакт-дисков

A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Распределенные ресурсы

Процесс  
Накопители  
на магнитных дисках  
Плоттеры  
Сканеры  
Устройства для чтения  
компакт-дисков

A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

E = (6342)  
P = (5322)  
A = (1020)

Ресурсы, которые еще нужны

# Предотвращение взаимоблокировки

---

Нарушаем «абсолютность» одного из следующих условий:

1. Условие взаимного исключения
2. Условие удержания и ожидания
3. Условие невыгружаемости
4. Условие циклического ожидания

# Блокировки

---

1. Оптимистические 😊
2. Пессимистические ☹️  
Двухфазная блокировка
3. При обмене данными
4. Активные
5. Зависание (НЕ БЛОКИРОВКА!!!)

# Планирование процессов

---

1. ЧТО?
2. ЗАЧЕМ?
3. КОГДА?

# Виды процессов

---

1. Ограниченные скоростью вычислений
2. Ограниченные скоростью работы устройств ввода-вывода



# Когда планировать

---

1. Когда процессор стартует
2. Когда процесс завершается
3. Когда процесс блокируется
4. Когда происходит прерывание ввода-вывода

# Виды планирования процессов

---

## 1. Все системы

- Равнодоступность
- Баланс

## 2. Пакетные системы

- Производительность
- Обратное время
- Максимальное использование центрального процессора

# Виды планирования процессов

---

## 3. Интерактивные системы

- Время отклика
- Пропорциональность

## 4. Системы реального времени

- Соблюдение предельных сроков
- Предсказуемость

## 5. Мультипроцессорные системы

# Планирование в пакетных системах

---

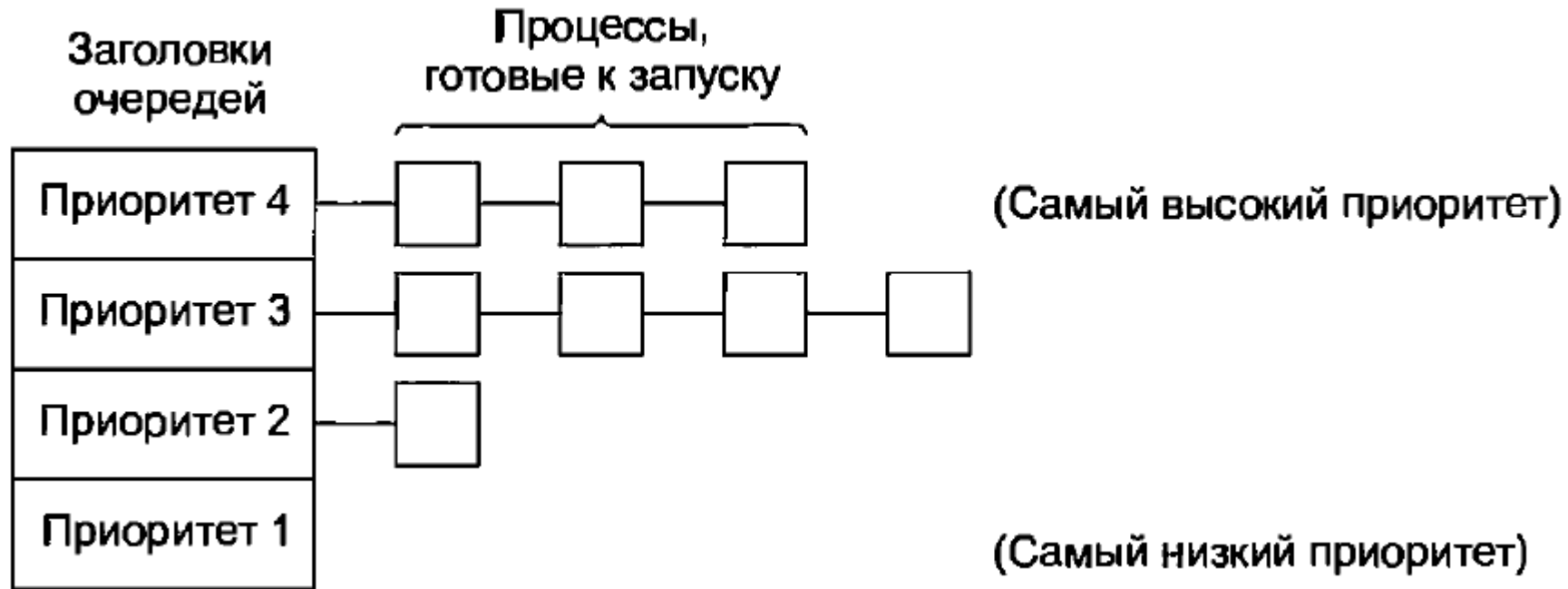
1. Первым пришел-первым вышел
2. Первое – самое короткое задание
3. Выполнение по приоритету наименьшего времени

# Планирование в интерактивных системах

---

1. Циклическое планирование
2. Приоритетное планирование
3. Использование нескольких очередей
4. Выбор следующего самого короткого процесса
5. Гарантированное планирование
6. Лотерейное планирование
7. Справедливое планирование

# Использование нескольких очередей



# Выбор следующего самого короткого

---

1. Вопрос в метрике определения самого короткого
2. Можно использовать
  - $(T_1 + \dots + T_N)/N$
3. Можно использовать
  - $T_0$
  - $T_0/2 + T_1/2$
  - $T_0/4 + T_1/4 + T_2/2$
  - $T_0/8 + T_1/8 + T_2/4 + T_3/2$

# Гарантированное планирование

---

1. Каждому дается  $K = X/N$  времени
  - $X$  – суммарное время работы процессора
  - $N$  – количество процессоров
2. Для каждого процесса высчитывается  $M = U/K$ 
  - $U$  – использованное время процессом
3. Когда наступает момент выбора следующего процесса, то выбирается тот, кто имеет минимальное  $M$
4. Время работы процесса определяется пока его  $M$  не превысила  $M$  ближайшего конкурента



# Лотерейное планирование

---

1. У  $i$ -го процесса есть  $K_i$  лотерейных билетов
2. Сумма всех лотерейных билетов  $K = \sum_1^N K_i$
3. При выдаче кванта времени выбирается случайно один билет
4. Билеты процессы могут передавать между собой

# Справедливое планирование

---

1. Если система становится многопользовательской
2. Комбинирование предыдущих методов планирования + разбиение на пользователей
3. Пусть есть 3 пользователя с соотношением выделенного времени 50%, 25%, 25% соответственно
  - Лотерейное планирование среди процессов пользователя 1
  - Лотерейное планирование среди процессов пользователя 2
  - Лотерейное планирование среди процессов пользователя 1
  - Лотерейное планирование среди процессов пользователя 3

# Планирование в системах реального времени

---

1. Алгоритм планирования RMS
2. Алгоритм планирования EDF

# Характеристики планирования в режиме реального времени

---

1. Гибкие/жесткие системы реального времени

2. Система называется планируемой, если:

$$\sum_{i=1}^m \frac{C_i}{P_i} \leq 1$$

3. С – время выполнения задания, Р – период его выполнения

# RMS – (Rate Monotonic Scheduling)

---

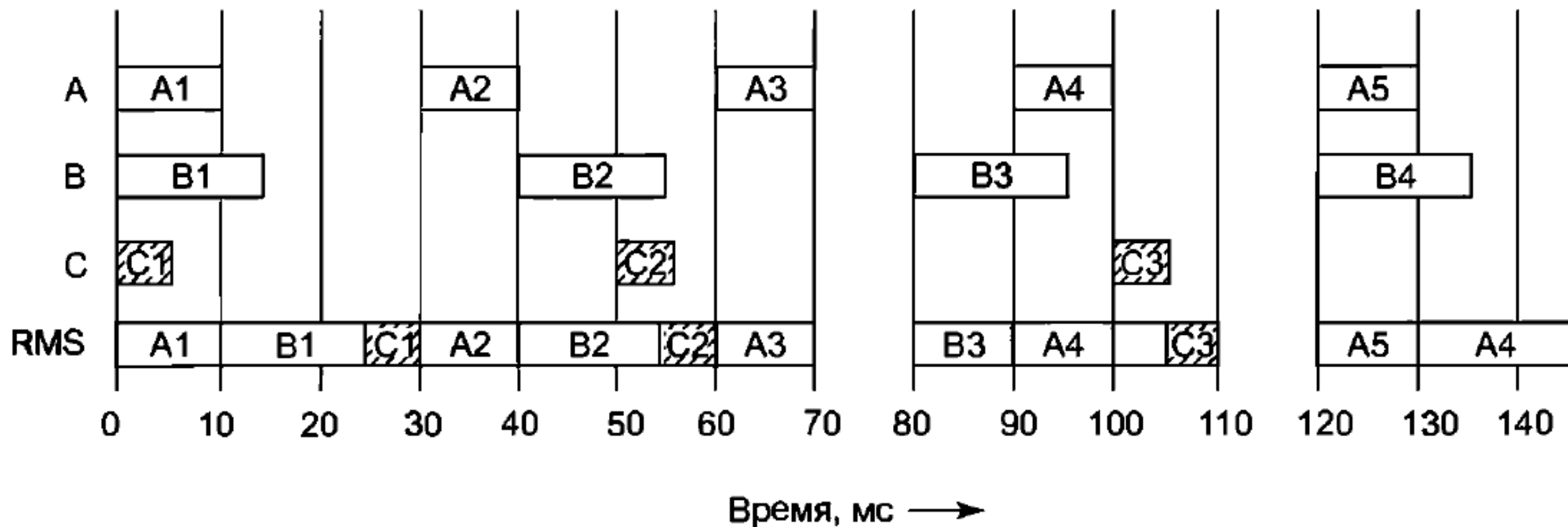
1. Статический алгоритм планирования
2. Приоритеты задаются в зависимости от частоты вызова задач
3. Рамки применимости
  - Каждый периодический процесс должен быть завершен в определенный срок
  - Процессы независимы друг от друга
  - Константное выполнение каждого процесса
  - У непериодических процессов нет крайних сроков
  - Вытеснение процесса происходит моментально

# Принцип работы RMS

---

1. Для периодических процессов считаем частоты их выполнения
  - $P = 25\text{мс} \Rightarrow \text{Частота} = 40$
2. Частота и является приоритетом для процесса
3. Когда необходимо выполнить периодический процесс, то он прерывается все процессы с меньшим приоритетом

# Пример RMS



# EDF (Earliest Deadline First)

---

1. Динамический алгоритм планирования реального времени
2. Приоритет назначается в зависимости от оставшегося крайнего срока выполнения

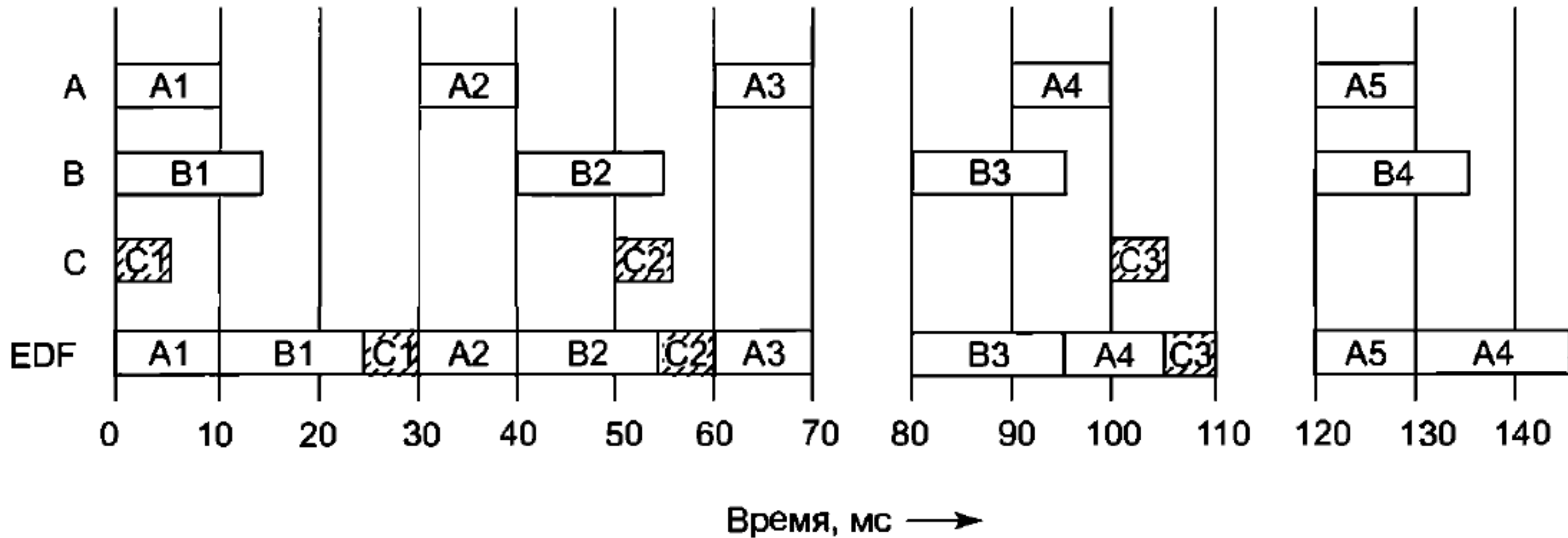


# Принцип работы EDF

---

1. Планировщик ведет список готовых процессов, отсортированный по крайним срокам выполнения
2. В работу всегда берется процесс с самым близким крайним сроком выполнения
3. При поступлении нового процесса на выполнение происходит перепланировка

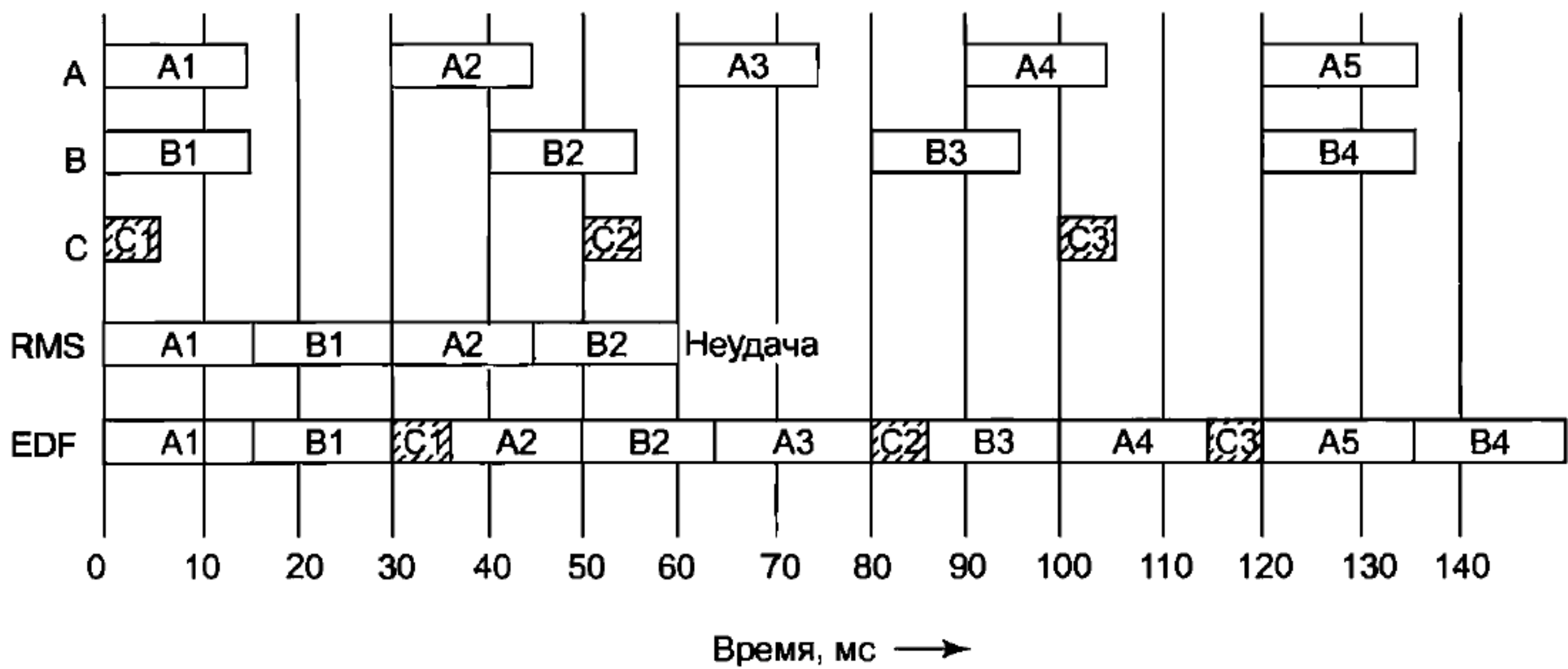
# Пример выполнения EDF



# Разница EDF, RMS

---

1. Для EDF требуются накладные расходы на динамическую приоритезацию
2. Для RMS есть рамки применимости:
  - $\sum_{i=1}^m \frac{c_i}{p_i} \leq m(2^{\frac{1}{m}} - 1)$

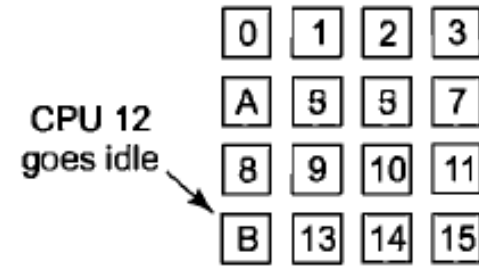
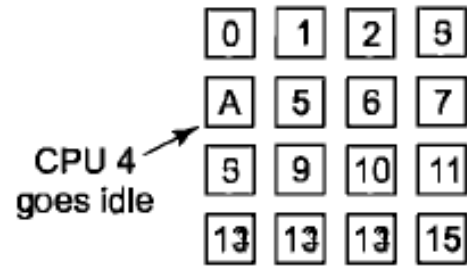
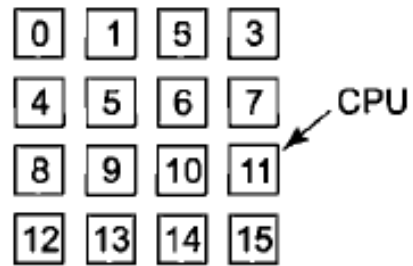


# Планирование в мультипроцессорных системах

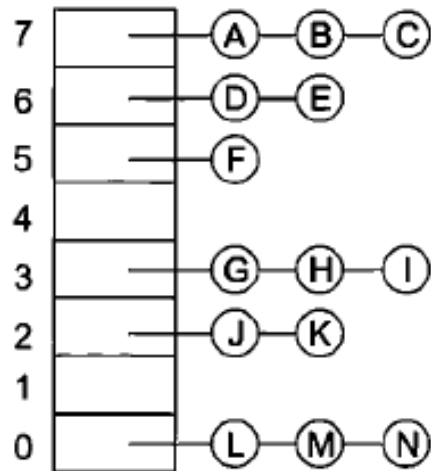
---

1. Потоки/процессы
2. Процессоры
3. Группы потоков

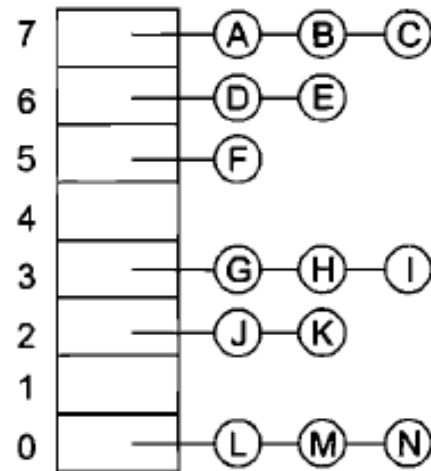
# Разделение времени (независимые потоки)



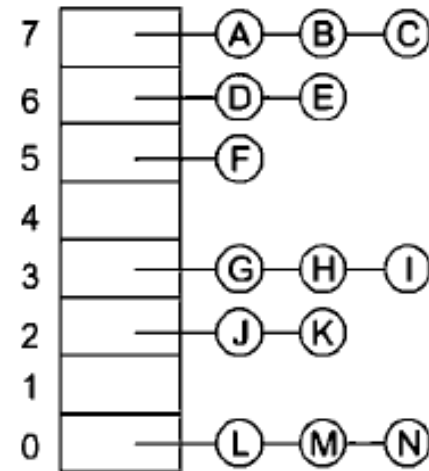
Приоритет



Приоритет



Приоритет



# Разделение времени (улучшения)

---

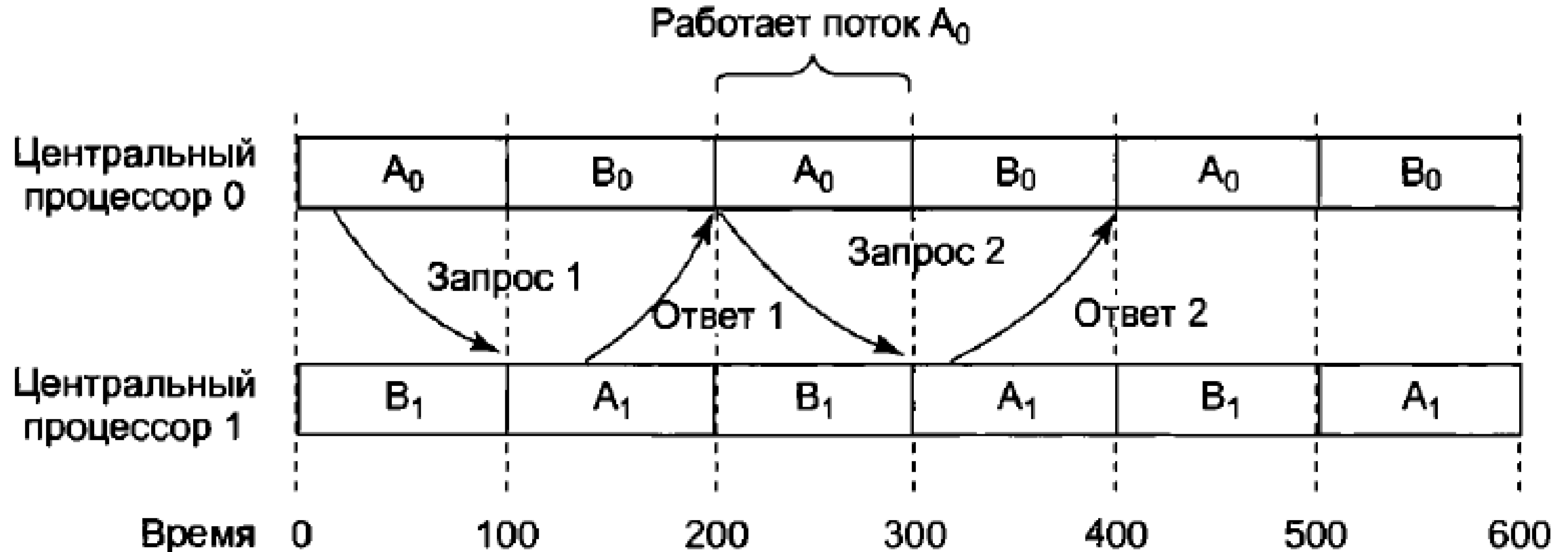
1. Разумное планирование  
spin lock
2. Родственное планирование (двухуровневый алгоритм планирования)

# Совместное использование пространства





# Проблемы совместного использования пространства



# Бригадное планирование

---

1. Группа взаимосвязанных потоков планируется совместно
2. Все члены бригады запускаются одновременно, на разных центральных процессорах, работающих в режиме разделения времени
3. У всех членов бригады есть кванты времени начинающиеся и заканчивающиеся одновременно

# Бригадное планирование

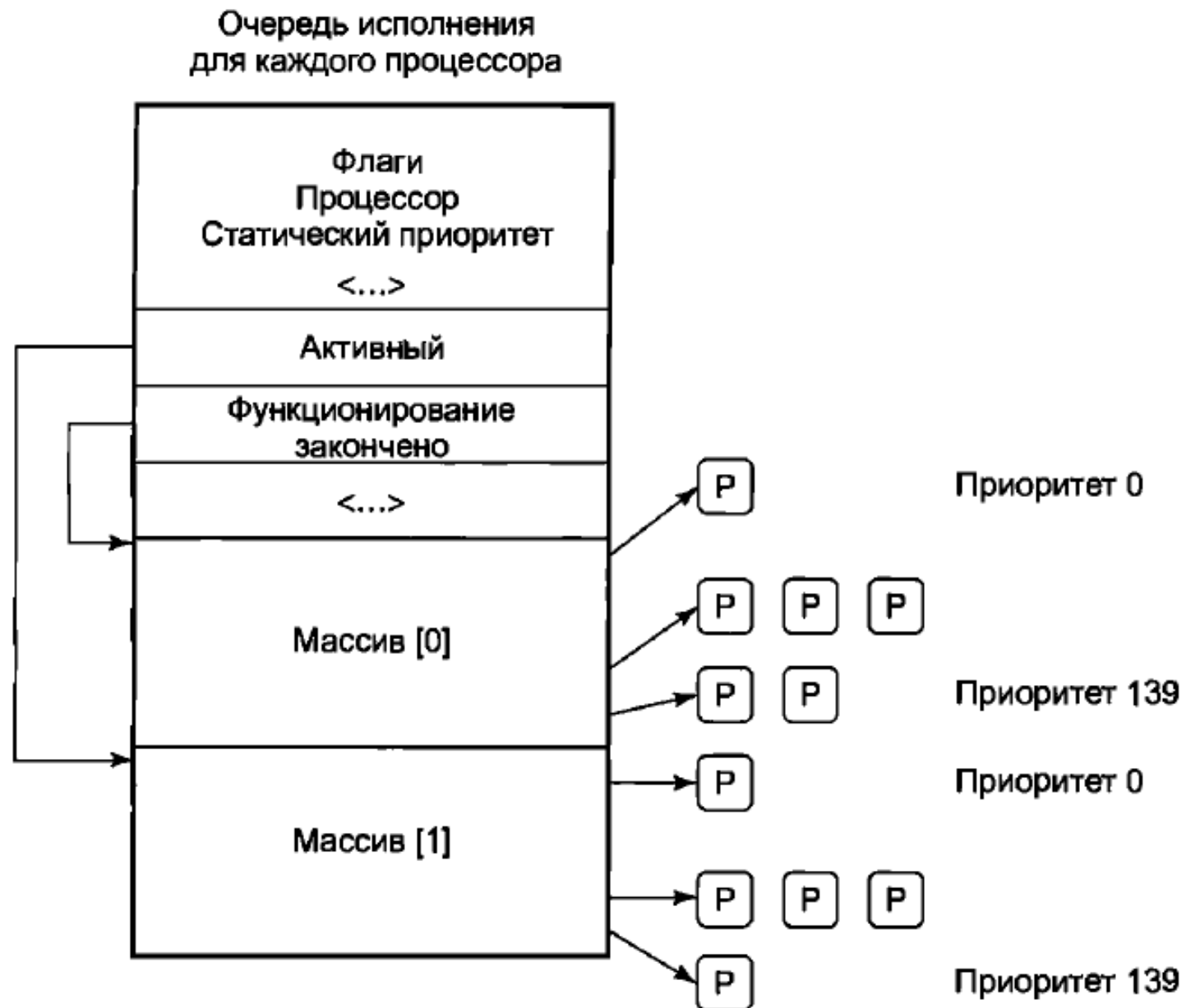
		Центральный процессор					
		0	1	2	3	4	5
Временной интервал	0	A <sub>0</sub>	A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>	A <sub>4</sub>	A <sub>5</sub>
	1	B <sub>0</sub>	B <sub>1</sub>	B <sub>2</sub>	C <sub>0</sub>	C <sub>1</sub>	C <sub>2</sub>
	2	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	E <sub>0</sub>
	3	E <sub>1</sub>	E <sub>2</sub>	E <sub>3</sub>	E <sub>4</sub>	E <sub>5</sub>	E <sub>6</sub>
	4	A <sub>0</sub>	A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>	A <sub>4</sub>	A <sub>5</sub>
	5	B <sub>0</sub>	B <sub>1</sub>	B <sub>2</sub>	C <sub>0</sub>	C <sub>1</sub>	C <sub>2</sub>
	6	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	E <sub>0</sub>
	7	E <sub>1</sub>	E <sub>2</sub>	E <sub>3</sub>	E <sub>4</sub>	E <sub>5</sub>	E <sub>6</sub>

# Классы приоритетов Linux

---

1. Потоки реального времени, обслуживание по алгоритму FIFO
2. Потоки реального времени обслуживание в порядке циклической очереди
3. Потоки разделения времени

# Очередь исполнения



# Особенности

---

1. Увеличение и уменьшение приоритета  $[-5; +5]$
2. Поощрение интерактивных процессов
3. Наказание «пожирателей» ресурсов
4. В зависимости от приоритета разная длина квантов времени

# События, инициализирующие планирование в Windows

---

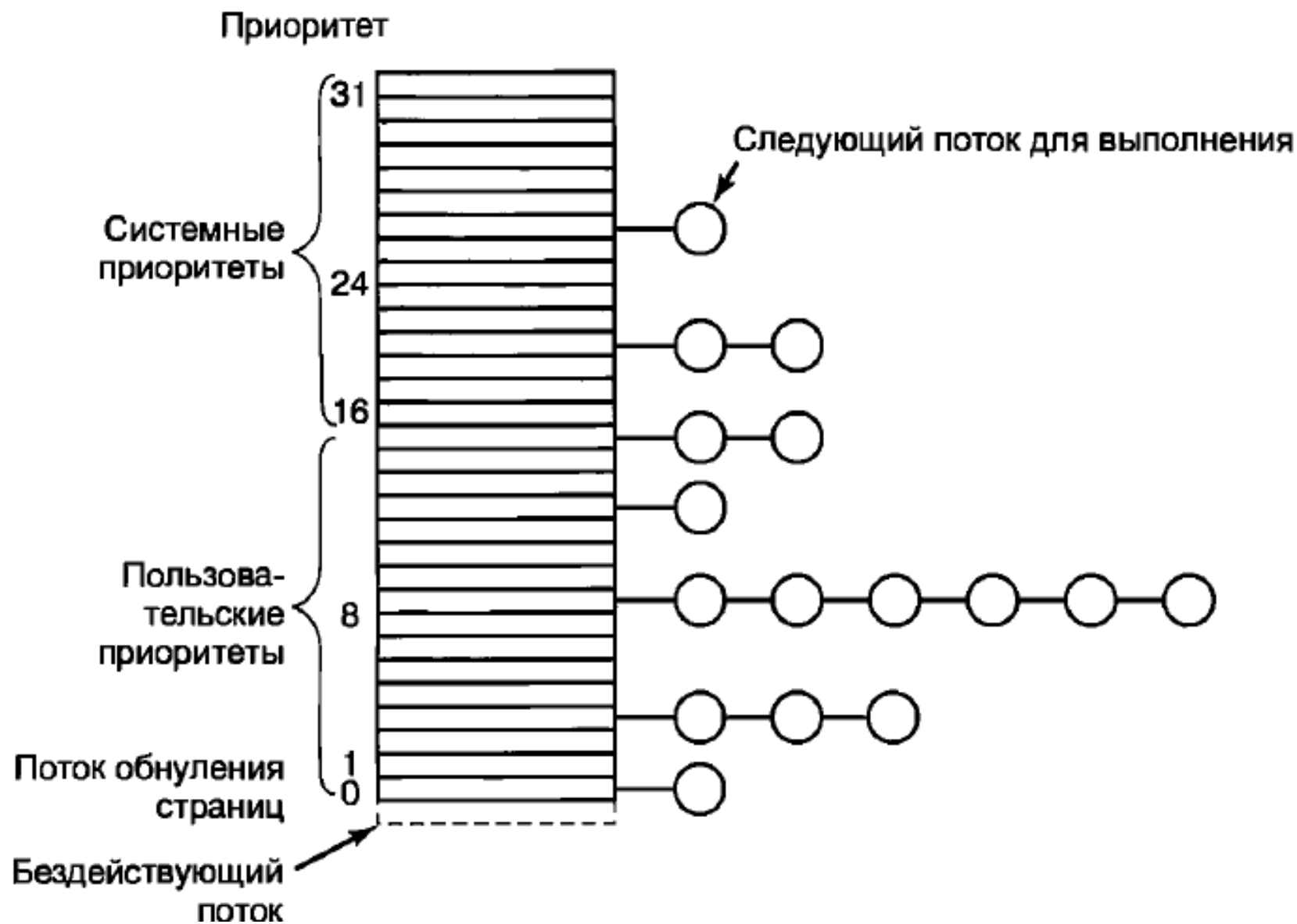
1. Истек квант времени
2. Поток блокируется на мьютексе, семафоре и тд
3. Поток сигнализирует о освобождение мьютекса, семафора и тд
4. Завершается операция ввода-вывода
5. Истекает время ожидания

# Классы приоритетов Windows

Классы приоритетов процессов Win32						
Приоритеты потоков Win32	Real-time	High	Above Normal	Normal	Below Normal	Idle
Time critical	31	15	15	15	15	15
Highest	26	15	12	10	8	6
Above normal	25	14	11	9	7	5
Normal	24	13	10	8	6	4
Below normal	23	12	9	7	5	3
Lowest	22	11	8	6	4	2
Idle	16	1	1	1	1	1



# Очередь исполнения



# Особенности

---

1. Приоритет увеличивается при завершение ввода-вывода  
диск – 1, клавиатура- 6, звуковая карта – 8
2. Повышение приоритета при освобождение мьютекса
3. При использование кванта времени поток падает на 1 уровень (и так до базового)
4. Если долго не выполнялся какой-то поток, то ему дают приоритет 15 на время 2-х квантов
5. Потоки, относящиеся к новому окну переднего плана получают удлинённый квант
6. Квант фиксирован (от 20 до 180 мс)

# Проблема инверсии приоритетов

---

