Операционные системы и сети ЭВМ

Operating Systems and Networking

Лекция 11

Сафонов Владимир Олегович,
профессор кафедры информатики,
руководитель лаборатории
Java-технологии
НИИММ СПбГУ
Email: **v_o_safonov@mail.ru**

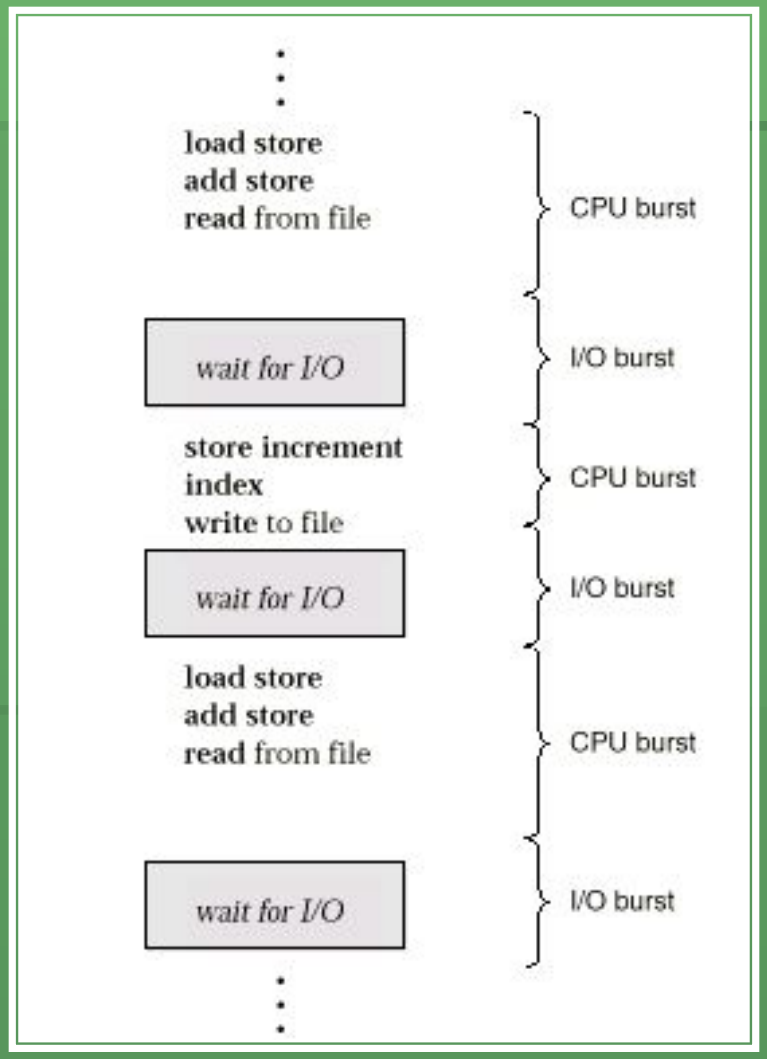
Планирование и диспетчеризация процессора

- **Основные понятия**
- **Критерии диспетчеризации**
- **Алгоритмы диспетчеризации**
- **Диспетчеризация нескольких процессоров**
- **Диспетчеризация в реальном времени**
- **Оценка алгоритма**

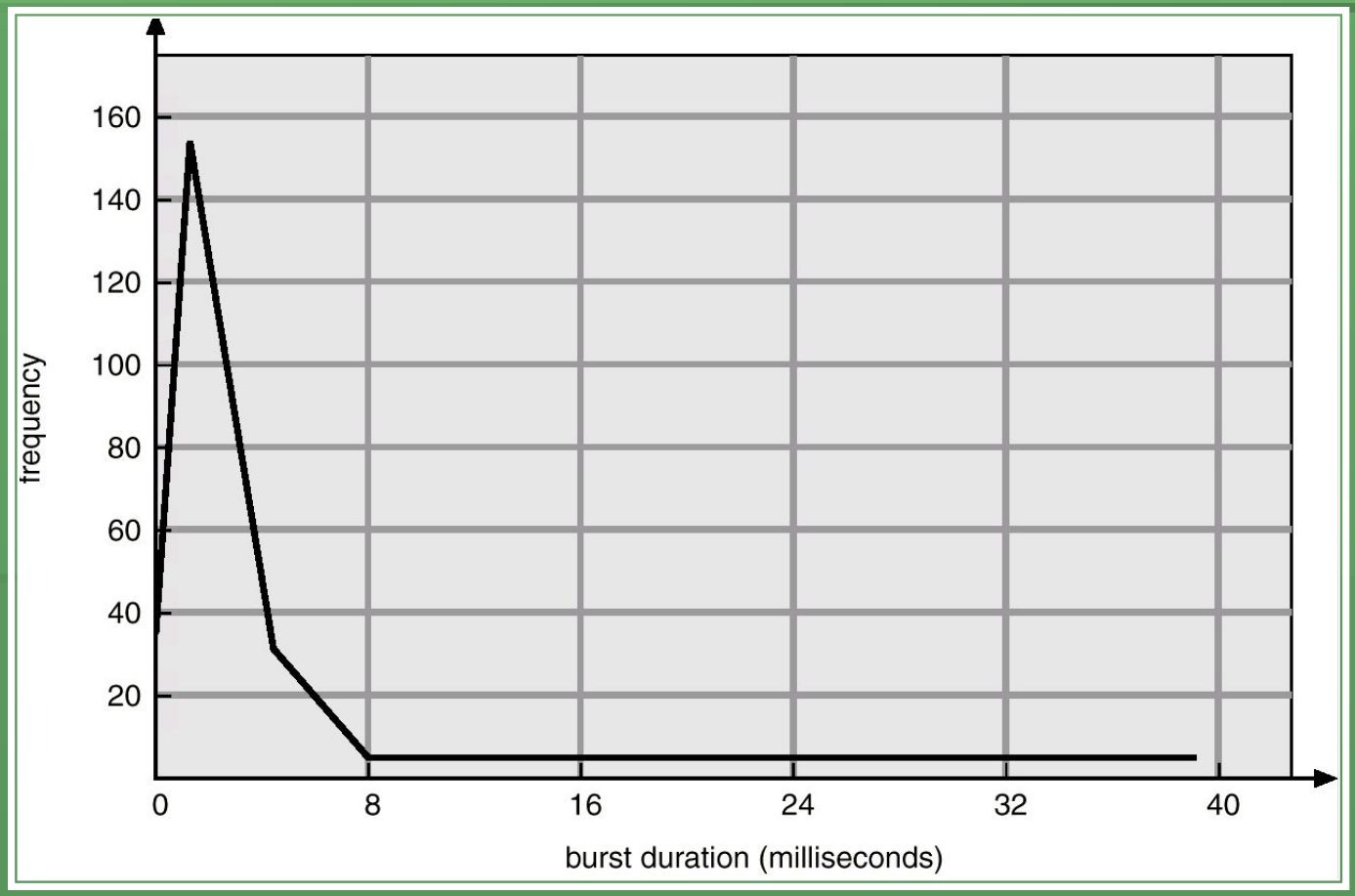
Основные понятия

- **Цель – максимальная загрузка процессора. Достигается п помощью мультипрограммирования**
- **Цикл CPU / I-O – Исполнение процесса состоит из работы процессора и ожидания ввода-вывода.**
- **Распределение периодов активности процессора**

Последовательность активных фаз **(bursts)** процессора и ввода-вывода



Гистограмма периодов активности процессора



Планировщик процессора **(scheduler)**

- **Выбирает один из нескольких процессов, загруженных в память и готовых к выполнению, и выделяет процессор для одного из них.**
- **Решения по диспетчеризации могут быть приняты, когда процесс:**
  1. **Переключается из состояния выполнения в состояние ожидания.**
  2. **Переключается из состояния выполнения в состояние готовности к выполнению.**
  3. **Переключается из состояния ожидания в состояние готовности.**
  4. **Завершается.**
- **Диспетчеризация типов 1 и 4 – *не опережающая (non-preemptive)*.**
- **В остальных случаях – *опережающая (preemptive).***

Собственно диспетчер

- **Модуль диспетчера предоставляет процессор тому процессу, который был выбран scheduler'ом, то есть:**
  - **Переключает контекст**
    - **Переключает процессор в пользовательский режим**
    - **Выполняет переход по соответствующему адресу в пользовательскую программу для ее рестарта**
- ***Dispatch latency* – время, требуемое для диспетчера, чтобы остановить один процесс и стартовать другой.**

Критерии диспетчеризации

- **Использование процессора – поддержание его в режиме занятости, насколько это возможно**
- **Пропускная способность (throughput) – число процессов, завершающих свое выполнение за единицу времени**
- **Время обработки (turnaround time) – время, необходимое для исполнения какого-либо процесса**
- **Время ожидания (waiting time)– время, которое процесс ждет в очереди процессов, готовых к выполнению**
- **Время ответа (response time) – время, требуемое от момента первого запроса до первого ответа (для среды разделения времени)**

Критерии оптимизации

- **Max CPU utilization**
- **Max throughput**
- **Min turnaround time**
- **Min waiting time**
- **Min response time**

Стратегия диспетчеризации
First-Come-First-Served (FCFS)

Процесс	Период активности
P_1	24
P_2	3
P_3	3

- Пусть порядок процессов таков: P_1 , P_2 , P_3
  Диаграмма Ганта (Gantt Chart) для их распределения:

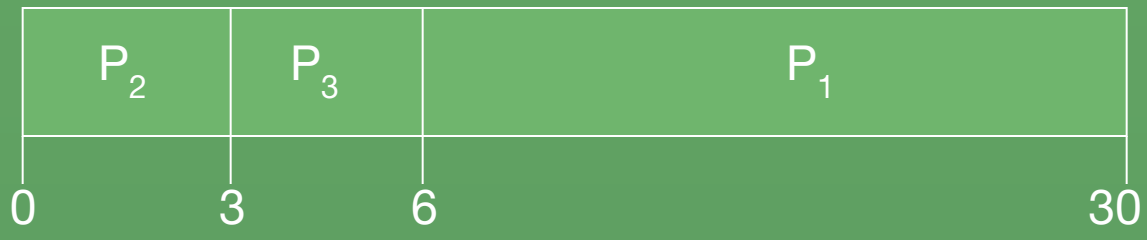
- Время ожидания для P_1 = 0; P_2 = 24; P_3 = 27
- Среднее время ожидания: (0 + 24 + 27)/3 = 17

Стратегия **FCFS** (продолжение)

Пусть порядок процессов таков:

P_2, P_3, P_1.

- **Диаграмма Ганта для их распределения:**



- Время ожидания: $P_1 = 6; P_2 = 0; P_3 = 3$
- Среднее время ожидания: $(6 + 0 + 3)/3 = 3$
- Много лучше, чем в предыдущем случае.
- *Эффект сопровождения (convoy effect)* - короткий процесс после долгого процесса

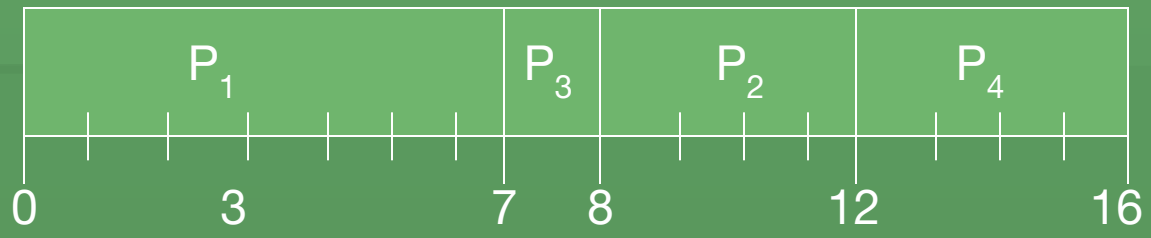
Стратегия **Shortest-Job-First (SJF)**

- **С каждым процессом связывается длина его очередного периода активности. Эта длина используется для того, чтобы первым обслужить самый короткий процесс .**
- **Две схемы:**
  - **Без опережения– пока процессу предоставляется процесс, он не может быть прерван, пока не истечет его кварнт времени.**
  - **С опережением – если приходит новый процесс, время активности которого меньше, чем оставшееся время активного процесса, - прервать активный процесс. Эта схема известна под названием Shortest-Remaining-Time-First (SRTF).**
- **SJF оптимальна – обеспечивает минимальное среднее время ожидания для заданного набора процессов.**

Пример: **SJF** без опережения

Процесс	Время появления	Время активности
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (без опережения)

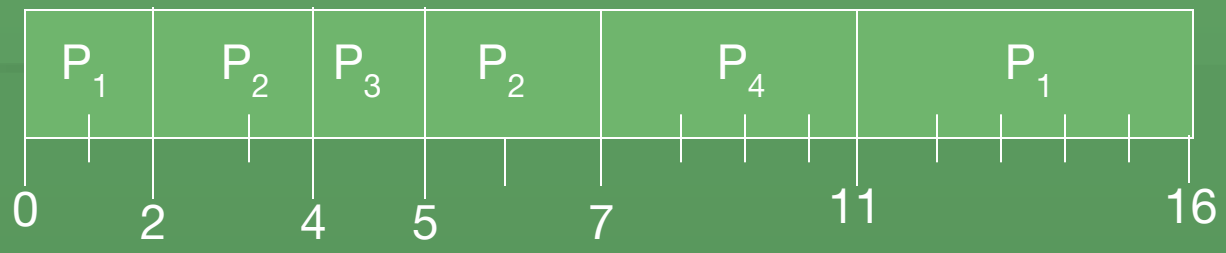


- Среднее время ожидания = (0 + 6 + 3 + 7)/4 = 4

Пример: **SJF** с опережением

Процесс	Время появления	Время активности
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- **SJF (с опережением)**



- **Среднее время ожидания = (9 + 1 + 0 +2)/4 = 3**

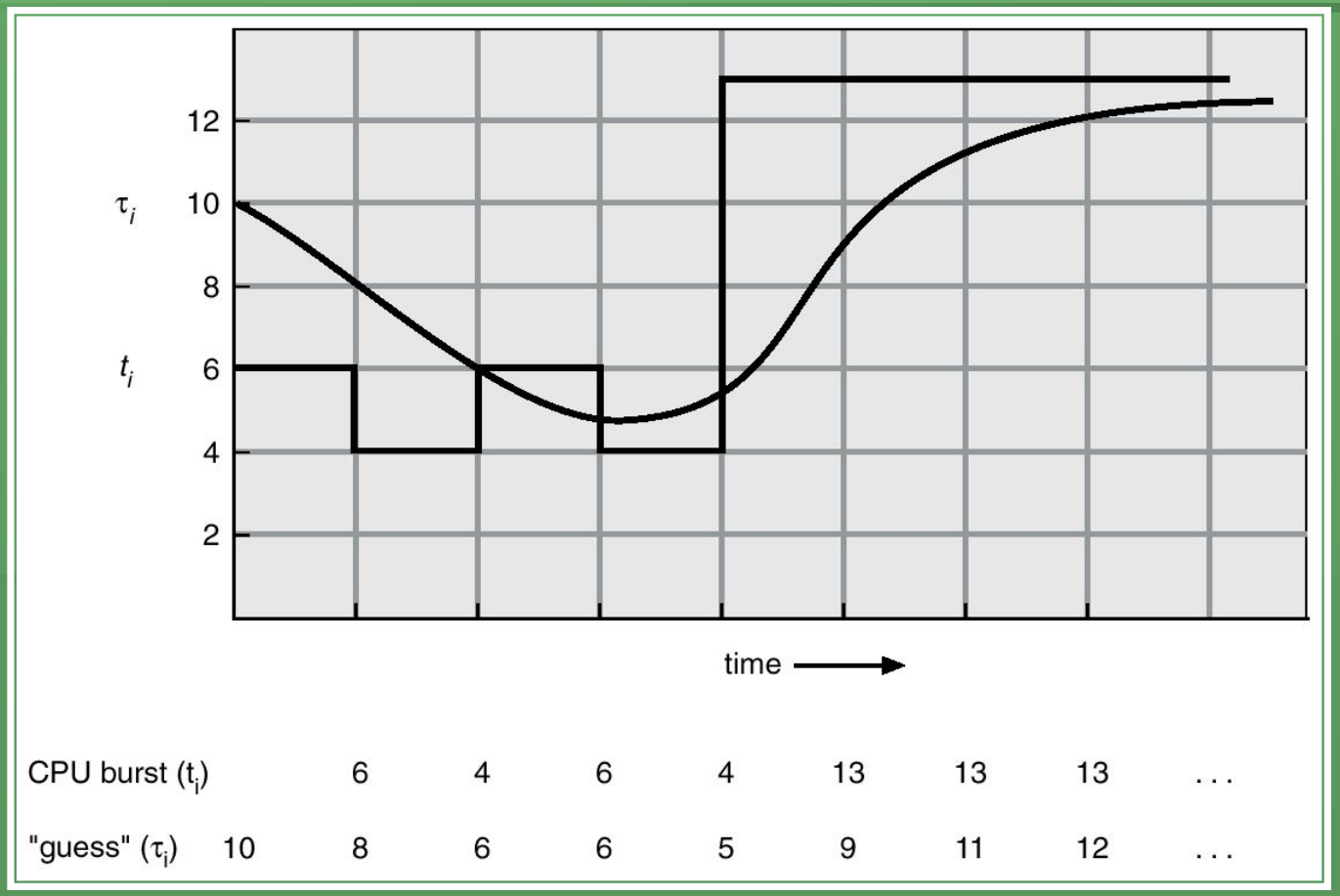
Определение длины следующего периода активности

- **Является лишь оценкой длины.**
- **Может быть выполнено с использованием длин предыдущих периодов активности, используя экспоненциальное усреднение**

1. t_n = actual lenght of n^{th} CPU burst
2. τ_{n+1} = predicted value for the next CPU burst
3. $\alpha, 0 \le \alpha \le 1$
4. Define:

$$\tau_{n=1} = \alpha\, t_n + (1-\alpha)\tau_n.$$

Предсказание длины следующего периода активности



Примеры экспоненциального усреднения

- **α =0**
  - **$\tau_{n+1} = \tau_n$**
  - **Недавняя история не учитывается.**
- **α =1**
  - **$\tau_{n+1} = t_n$**
  - **Учитывается только фактическая длина последнего периода активности.**
- **Если обобщить формулу, получим:**

$$\tau_{n+1} = \alpha t_n + (1 - \alpha)\,\alpha\, t_{n-1} + \ldots + (1 - \alpha)^j \alpha\, t_{n-1} + \ldots + (1 - \alpha)^{n+1} t_n\, \tau_0$$

- **Так как α и(1 - α) не превосходят 1, каждый последующий терм имеет меньший вес, чем его предшественник**

Диспетчеризация по приоритетам

- С каждым процессом связывается его приоритет (целое число)
- Процессор выделяется процессу с наивысшим приоритетом (меньшее число – высший приоритет)
- Стратегии с опережением и без опережения
- SJF – это диспетчеризация по приоритетам, в которой приоритетом является очередное время активности.
- Проблема ≡ “Starvation” (“голодание”) – процессы с низким приоритетом могут никогда не исполниться
- Решение ≡ “Aging” (“возраст”) – с течением времени приоритет процесса повышается.

Стратегия **Round Robin (RR)** – “круговая система”

- **Каждый процесс получает небольшой квант процессорного времени, обычно – 10-100 миллисекунд. После того, как это время закончено, процесс прерывается и помещается в конец очереди готовых процессов.**
- **Если всего *n* процессов в очереди готовых к выполнению, и квант времени - *q*, то каждый процесс получает 1/*n* процессорного времени порциями самое большее по *q* единиц за один раз. Ни один процесс не ждет больше, чем (*n*-1)*q* единиц времени.**
- **Производительность**

  ***q* велико ⇒ FIFO**

  ***q* мало ⇒ *q* должно быть бо́льшим, по сравнению со временем контекстного переключения, иначе слишком велики накладные расходы**

Пример **RR (**квант времени **= 20)**

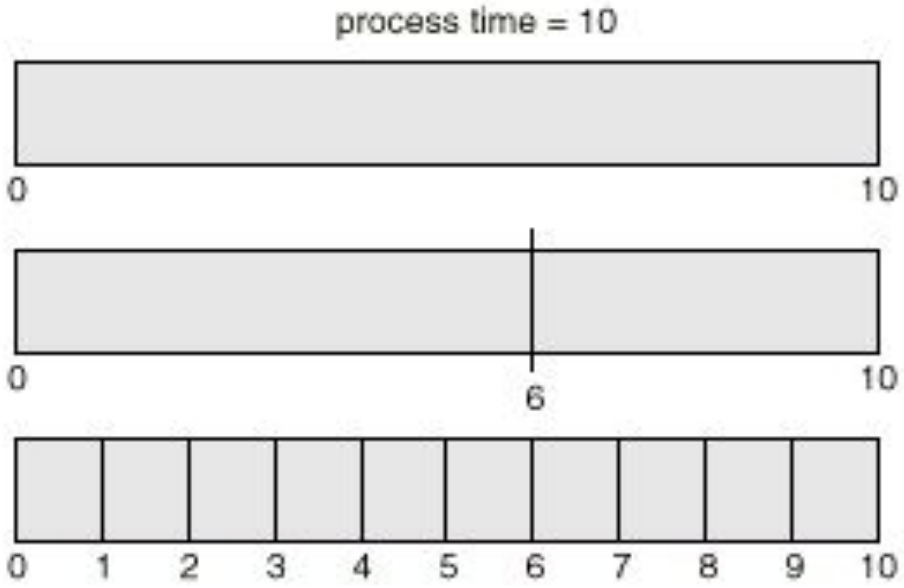
- **Пример RR с квантом времени = 20**

Процес	Время активности
P_1	53
P_2	17
P_3	68
P_4	24

- **Диаграмма Ганта:**

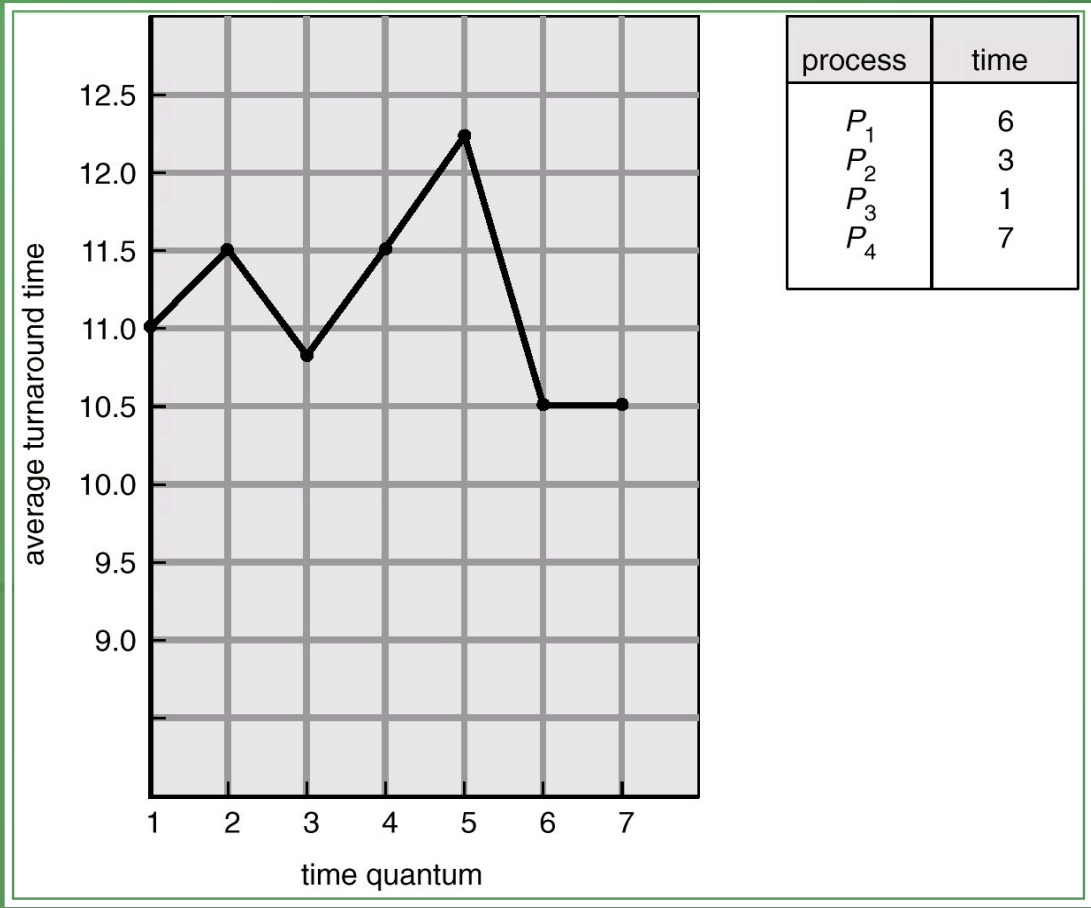
- **Обычно RR имеет худшее время оборота, чем SJF, но лучшее время ответа.**

Квант времени ЦП и время переключения контекста



quantum	context switches
12	0
6	1
1	9

Изменение времени оборота, в зависимости от кванта времени



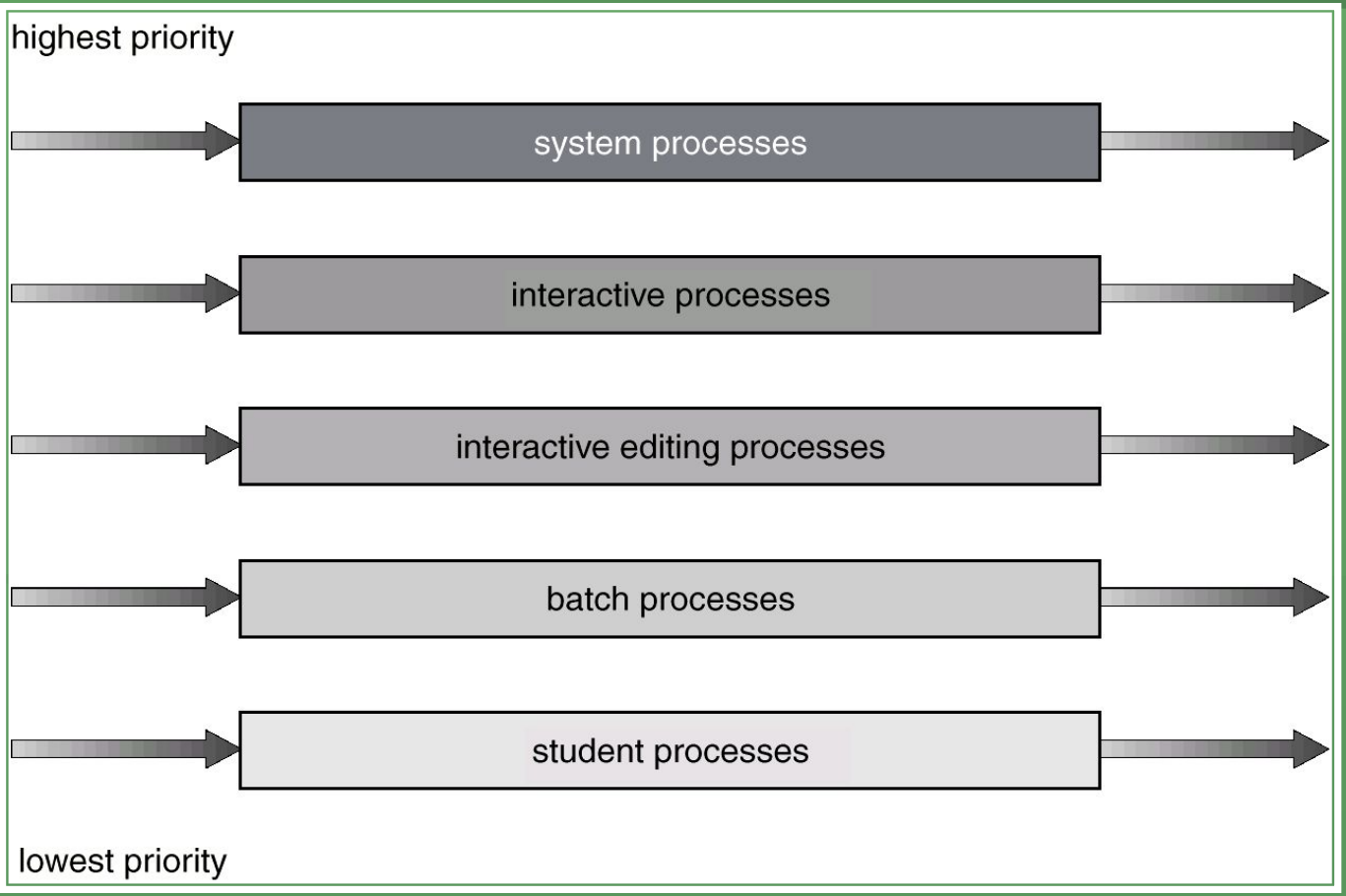
Многоуровневая очередь

- **Очередь готовых к выполнению процессов делится на две очереди:**

  основная (интерактивные процессы)
  фоновая (пакет)
- **Каждая очередь имеет свой собственный алгоритм диспетчеризации:**
  основная– RR
  фоновая – FCFS
- **Необходима также диспетчеризация между очередями.**
  - **С фиксированным приоритетом; (обслуживание всех процессов из основной очереди, затем – из фоновой). Есть вероятность “голодания”.**

    Выделение отрезка времени – каждая очередь получает некоторый отрезок времени ЦП, который она может распределять между процессами; например, 80 % - на RR в основной очереди; 20% на FCFS в фоновой очереди

Диспетчеризация по принципу многоуровневой очереди



Q & A

- Вопросы и ответы