

# Процессы

- Понятие о процессах и мультизадачности
- Планирование процессов

# Мультизадачность

- Большинство программ для ЭВМ **не** могут выполняться на процессоре (ЦП) **непрерывно**: им периодически требуется взаимодействие с периферийными устройствами (жесткий диск, клавиатура, мышь, экран и т.д.).
- Работа с устройствами протекает значительно **медленнее**, чем вычисления на процессоре. Изначально процессор занимался передачей данных на устройство и приемом ответа.

# Мультизадачность

- С изобретением технологии *DMA* (Direct Memory Access) устройства смогли работать с оперативной памятью напрямую, минуя процессор.
- Пока программа ожидает ответа устройства, работающего через DMA процессор простаивает. В это время его можно использовать для выполнения других программ.
- Выполняемая программа получила название **процесса**.

# Процесс

- **Процессом** называют программу, находящуюся в состоянии выполнения, с выделенными ей ресурсами (оперативной памятью, открытыми файлами и т.д.)

## Процесс $\neq$ Программа

- одну программу можно запустить несколько раз (параллельно)
- в некоторых ОС одна программа может состоять из нескольких процессов
- процесс может изменить выполняемую в нем программу на другую

# Невытесняющая мультизадачность

- При невытесняющей мультизадачности процесс сам решает, когда вернуть управление процессору, обычно это происходит в двух случаях:
  - процесс начал выполнять операцию на периферийном устройстве;
  - процесс сообщил ОС, что его можно прервать для выполнения других програм.
- Невытесняющая мультизадачность была реализована в Windows 3

# Невытесняющая мультизадачность

- Достоинства
  - простота реализации
- Недостатки
  - программист должен регулярно возвращать управление при выполнении длительных вычислений;
  - программа может злонамеренно захватить процессорное время, лишая другие процессы возможности выполниться;
  - бесконечное зацикливание вследствие ошибки в любой программе, если цикл содержит только вычисления, приведет к зависанию всей ОС

# Вытесняющая мультизадачность

- При вытесняющей мультизадачности, если процесс не вернул управление в течении определенного срока (**кванта** процессорного времени), ОС принудительно приостанавливает его и заменяет другим.
- Это дает возможность ОС контролировать распределение процессорного времени, а также сохранять стабильность несмотря на ошибки в программах пользователей.

# Состояния процесса





# Состояния процесса

- Процесс находится в состоянии **ожидания**, когда он не может выполняться, поскольку ожидает какого-либо события (например ответа периферийного устройства).
- Процесс находится в состоянии **готовности**, если он может выполняться, но процессор занят другим процессом.

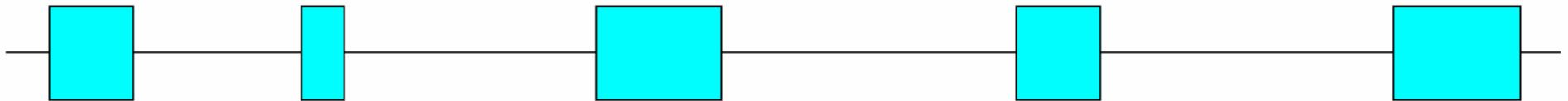
# Планирование процессов

Процесс, зависящий от процессора



CPU burst

Процесс, зависящий от устройств ввода/вывода



I/O burst

# Цели планирования

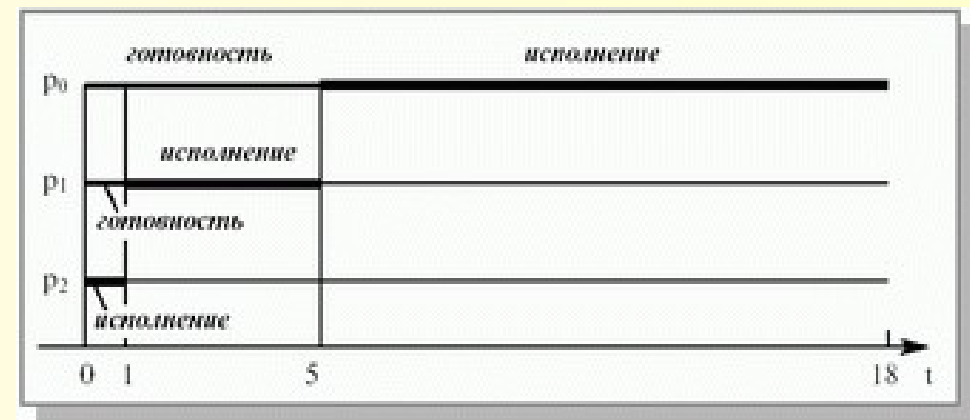
- **Справедливость** – гарантировать каждому заданию или процессу определенную часть времени использования процессора в компьютерной системе, стараясь не допустить возникновения ситуации, когда процесс одного пользователя постоянно занимает процессор, в то время как процесс другого пользователя фактически не начинал выполняться.
- **Эффективность** – постараться занять процессор на все 100% рабочего времени, не позволяя ему простаивать в ожидании процессов, готовых к исполнению. В реальных вычислительных системах загрузка процессора колеблется от 40 до 90%.

# Цели планирования

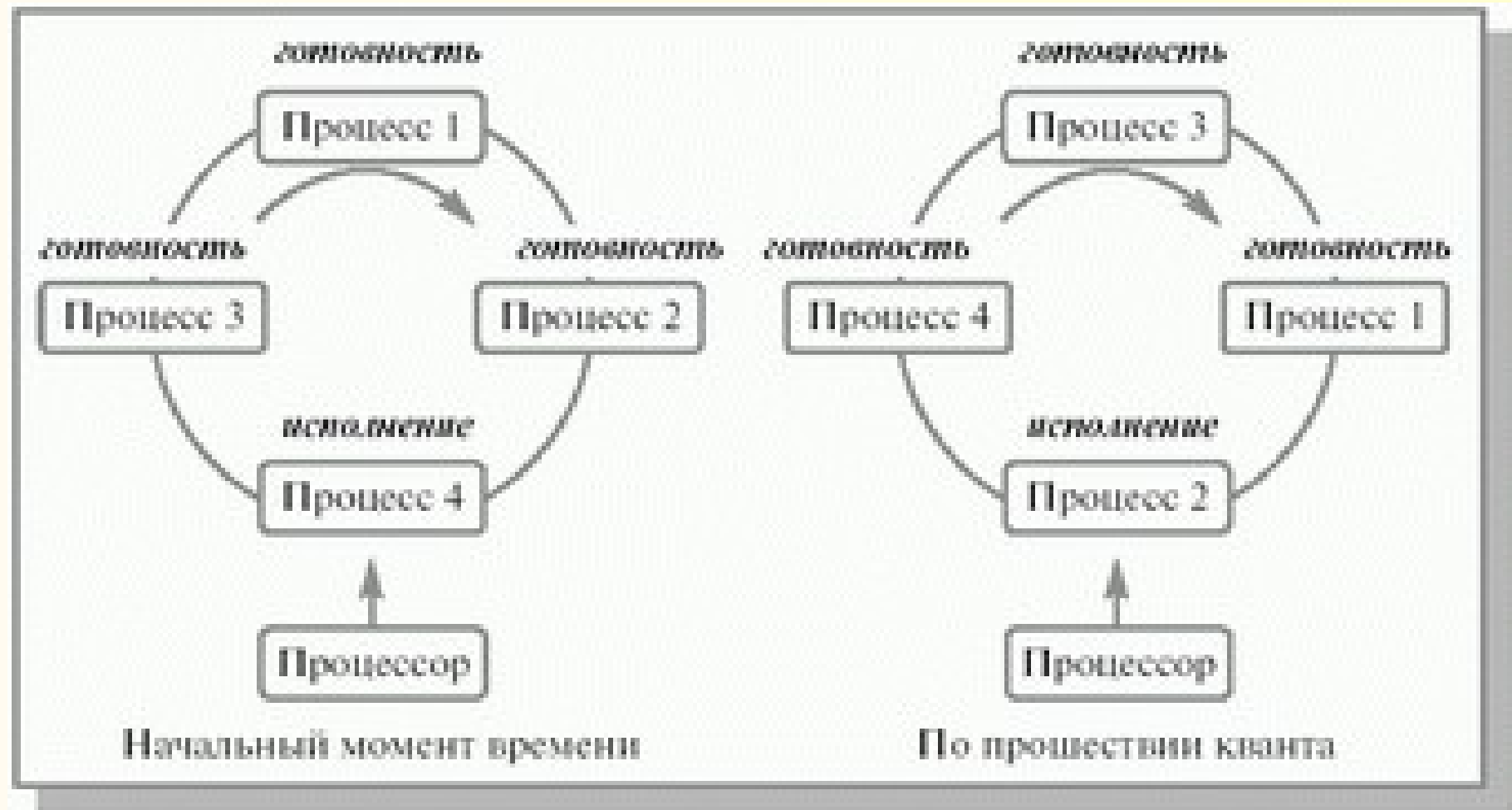
- Сокращение **полного времени выполнения** (turnaround time) – обеспечить минимальное время между стартом процесса или постановкой задания в очередь для загрузки и его завершением.
- Сокращение **времени ожидания** (waiting time) – сократить время, которое проводят процессы в состоянии готовности и задания в очереди для загрузки.
- Сокращение **времени отклика** (response time) – минимизировать время, которое требуется процессу в интерактивных системах для ответа на запрос пользователя.

# First-Come, First-Served (FCFS)

Процессы



# Round Robin



# Round Robin

- Вытесняющий вариант FCFS.
- Обеспечивает относительное равноправие процессов.
- Производительность зависит от выбора кванта времени.
- Слишком маленький квант увеличивает расходы времени на переключение процессов.
- При слишком большом кванте превращается в FCFS.

# Shortest-Job-First (SJF)

Идеальный случай, но сложно предсказать величину CPU burst.

Пусть  $t(n)$  – величина  $n$ -го CPU burst,  $T(n + 1)$  – предсказываемое значение для  $n + 1$ -го CPU burst,  $a$  – некоторая величина в диапазоне от 0 до 1.

Определим рекуррентное соотношение

$$T(n+1) = at(n) + (1-a)T(n)$$

$T(0)$  положим произвольной константой. Первое слагаемое учитывает последнее поведение процесса, тогда как второе слагаемое учитывает его предысторию.

Обычно выбирают  $a=1/2$ . Так удобнее и считать, т.к. деление на 2 осуществляется побитовым сдвигом.



# Справедливое планирование

- **Гарантированное выполнение.** Пронумеруем всех пользователей от 1 до  $N$ . Для каждого пользователя с номером  $i$  введем две величины:  $T_i$  – время нахождения пользователя в системе или, другими словами, длительность сеанса его общения с машиной и  $t_i$  – суммарное процессорное время уже выделенное всем его процессам в течение сеанса.
- Справедливым для пользователя было бы получение  $T_i/N$  процессорного времени. Вычисляется для процессов каждого пользователя значение коэффициента справедливости  $t_i * N / T_i$  и очередной квант времени предоставляется готовому процессу с наименьшей величиной этого отношения.

# Справедливое планирование

- **Статистическая справедливость.** Каждому процессу выделяется лотерейный билет (приоритетным процессам может выделяться несколько), система случайно выбирает билет и запускает на выполнение соответствующий процесс. Поскольку переключение задач происходит очень часто (в современных системах в среднем 20-60 раз в секунду), каждый процесс получает долю процессорного времени в соответствии со своим приоритетом. Этот алгоритм обладает высокой скоростью из-за простоты.

# Приоритетные очереди

