

Планирование загрузки центрального процессора

Операционные системы. Лекция 3
Павенко Е.Н., НГТУ

Уровни планирования процессов

- Долгосрочное планирование – планирование заданий.
- Среднесрочное планирование – swapping.
- Краткосрочное планирование – планирование использования процессора.

Цели планирования

- Справедливость
- Эффективность
- Сокращение полного времени выполнения (turnaround time)
- Сокращение времени ожидания (waiting time)
- Сокращение времени отклика (response time)

Желаемые свойства алгоритмов планирования

- Предсказуемость
- Минимизация накладных расходов.
- Равномерность загрузки вычислительной системы.
- Масштабируемость.

Параметры планирования

- Статические параметры вычислительной системы – например, предельные значения ее ресурсов.
- Статические параметры процесса – кем запущен, степень важности, запрошенное процессорное время, какие требуются ресурсы и т.д.

статические

- Динамические параметры вычислительной системы – например, количество свободных ресурсов в данный момент.
- Динамические параметры процесса – текущий приоритет, размер занимаемой оперативной памяти, использованное процессорное время и т.д.

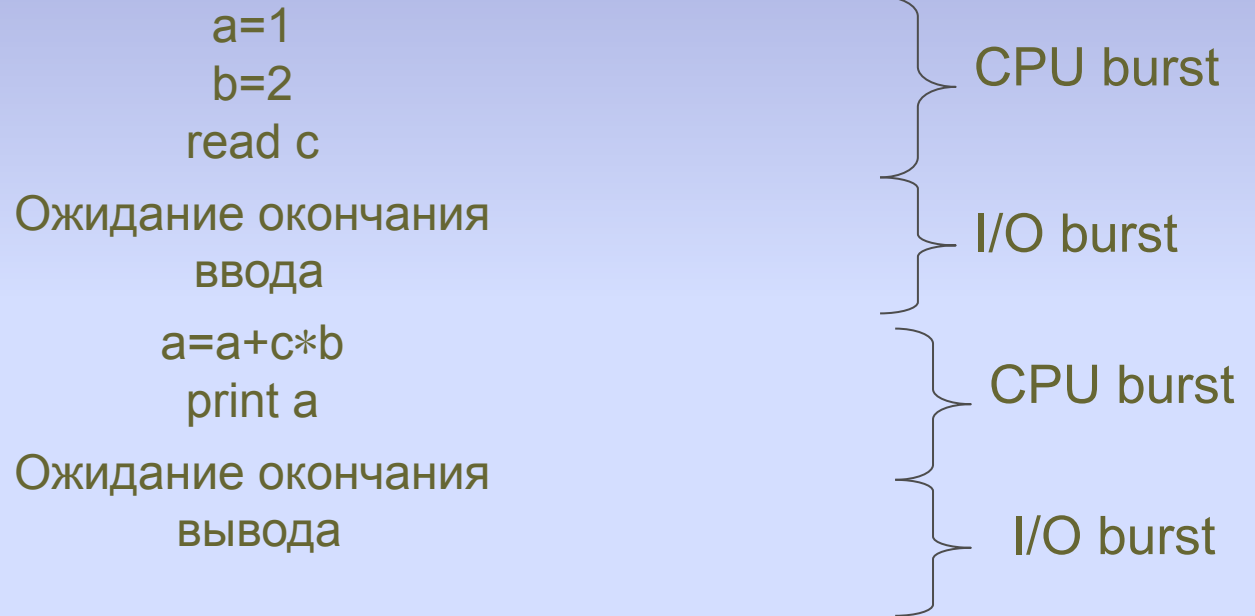
динамические

Параметры планирования

- Долгосрочное планирование:
Статические и динамические параметры вычислительной системы и статические параметры процесса.
- Среднесрочное планирование:
Статические и динамические параметры вычислительной системы и статические и динамические параметры процесса.
- Краткосрочное планирование:
Статические и динамические параметры вычислительной системы, статические и динамические параметры процесса , CPU burst, I/O burst.

CPU burst и I/O burst

Важные динамические параметры процесса



Вытесняющее и невытесняющее планирование

1. Перевод процесса из состояния *исполнение* в состояние *закончил исполнение*
2. Перевод процесса из состояния *исполнение* в состояние *ожидание*

Вынужденное принятие решения

Принятие только вынужденных решений – невытесняющее планирование

3. Перевод процесса из состояния *исполнение* в состояние *готовность*
4. Перевод процесса из состояния *ожидание* в состояние *готовность*

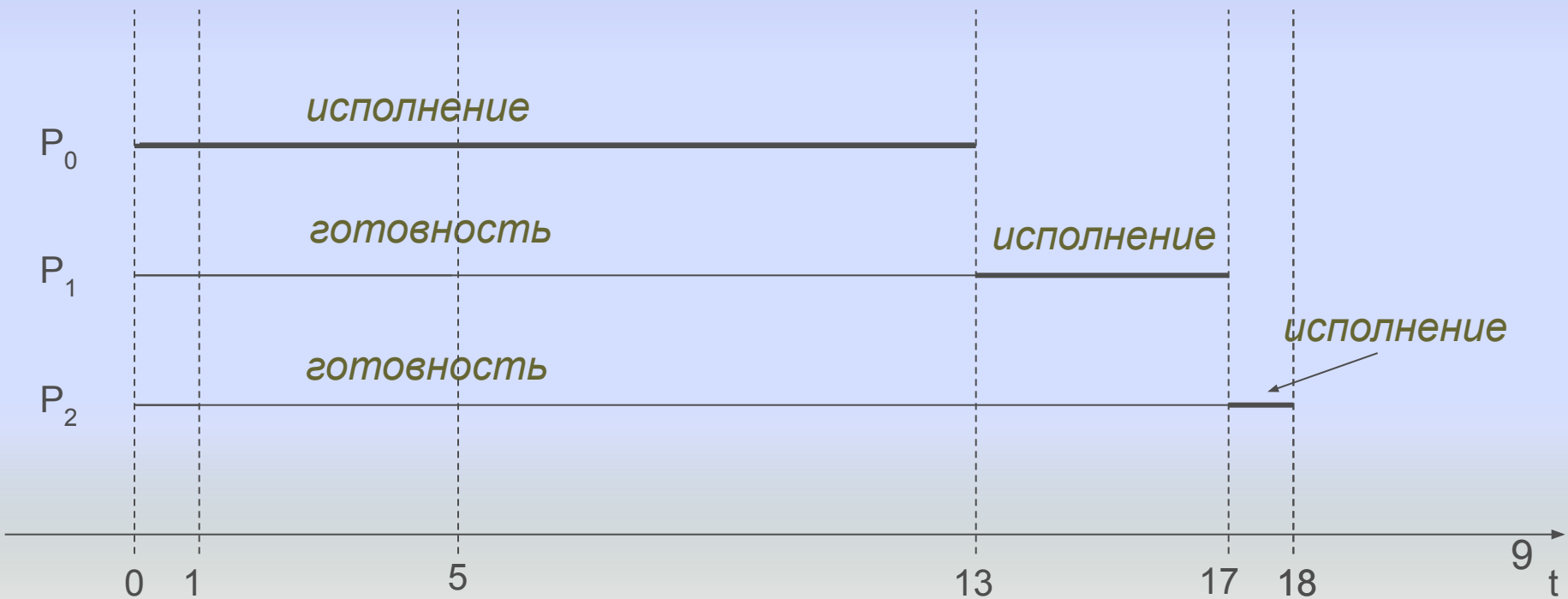
Невынужденное принятие решения

Принятие вынужденных и невынужденных решений – вытесняющее планирование

Алгоритмы планирования

FCFS (First Come – First Served)

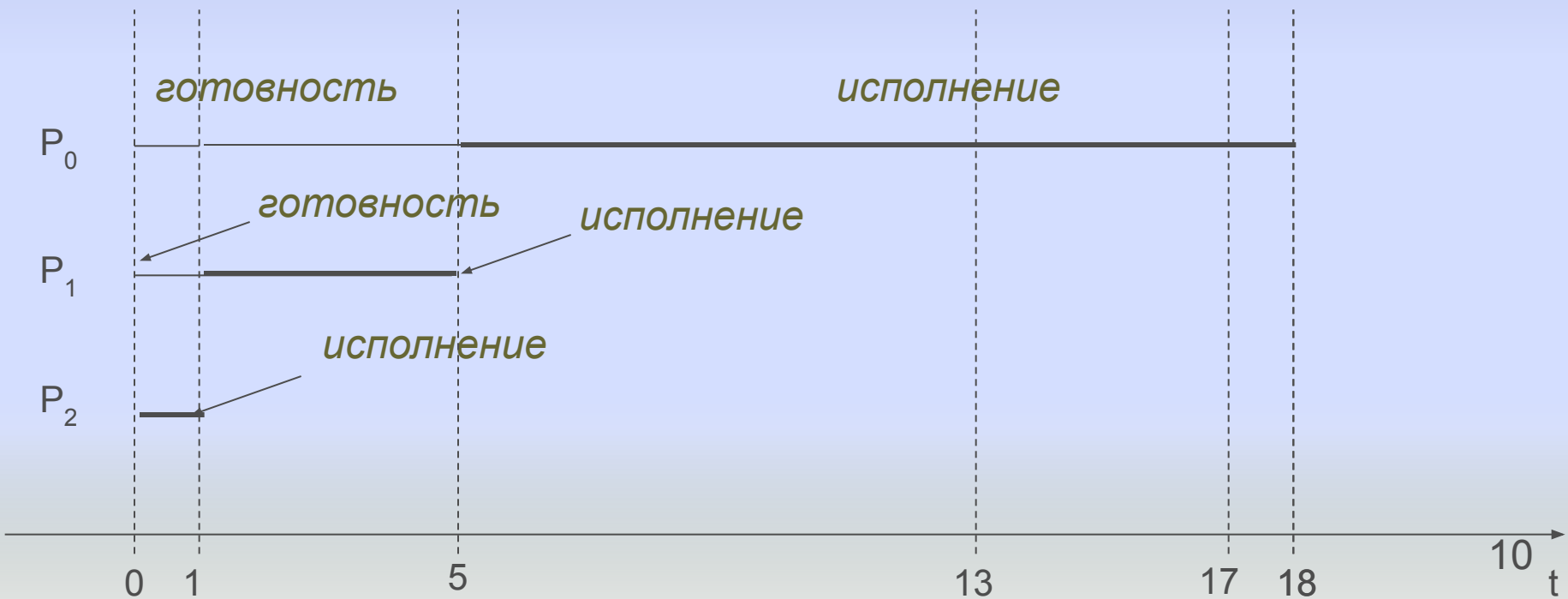
Процессы	P_0	P_1	P_2
Продолжительность CPU burst	13	4	1



Алгоритмы планирования

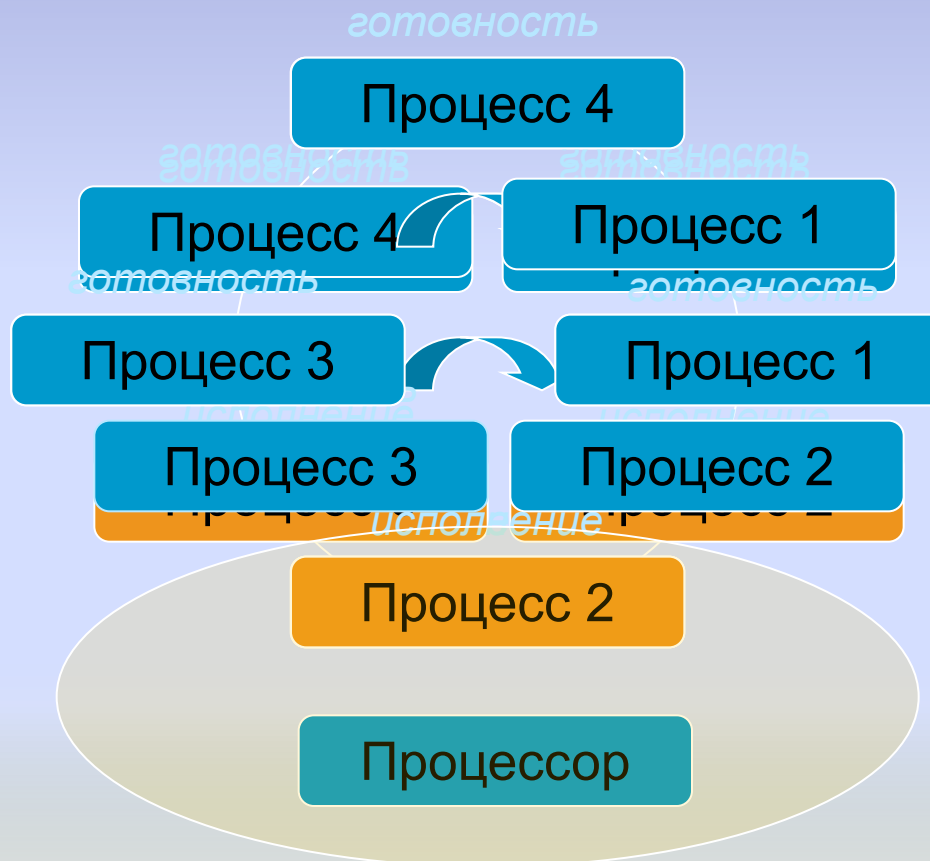
FCFS (First Come – First Served)

Процессы	P_2	P_1	P_0
Продолжительность CPU burst	1	4	13



Алгоритмы планирования

RR (Round Robin)



Алгоритмы планирования

RR (Round Robin)

- **Остаток времени CPU burst \leq кванта времени:**
 - процесс освобождает процессор до истечения кванта;
 - на исполнение выбираем новый процесс из начала очереди готовых;
- **Остаток времени CPU burst \geq кванта времени:**
 - По окончании кванта процесс помещается в конец очереди готовых к исполнению процессов;
 - на исполнение выбираем новый процесс из начала очереди готовых.

Алгоритмы планирования

RR (Round Robin)

Процессы	P_0	P_1	P_2
Продолжительность CPU burst	13	4	1

Величина кванта времени – 4

время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P_0	И	И	И	И	Г	Г	Г	Г	Г	И	И	И	И	И	И	И	И	И
P_1	Г	Г	Г	Г	И	И	И	И										
P_2	Г	Г	Г	Г	Г	Г	Г	Г	И									

исполнение

P_0

Очередь готовых

P_0	P_0	P_0
-------	-------	-------

Алгоритмы планирования

RR (Round Robin)

Процессы	P_0	P_1	P_2
Продолжительность CPU burst	13	4	1

Величина кванта времени – 1

время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P_0	И	Г	Г	И	Г	И	Г	И	Г	И	И	И	И	И	И	И	И	И
P_1	Г	И	Г	Г	И	Г	И	Г	И									
P_2	Г	Г	И															

исполнение

P_0

Очередь готовых

P_0	P_0	P_0
-------	-------	-------

Алгоритмы планирования

SJF (Shortest Job First) НЕВЫТЕСНЯЮЩИЙ

Процессы	P_0	P_1	P_2	P_3
Продолжительность CPU burst	5	3	7	1

время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
P_0	Г	Г	Г	Г	И	И	И	И	И							
P_1	Г	И	И	И												
P_2	Г	Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И	И	И	И
P_3	И															

исполнение

P_0

ГОТОВНОСТЬ

P_0	P_1	P_2	P_3
-------	-------	-------	-------

Алгоритмы планирования

SJF (Shortest Job First) ВЫТЕСНЯЮЩИЙ

Процессы	P_0	P_1	P_2	P_3
Продолжительность CPU burst	6	2	5	5
Момент появления в очереди	0	2	6	0

время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P_0	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И	И	И
P_1			И	И														
P_2							Г	И	И	И	И	И						
P_3	И	И	Г	Г	И	И	И											

исполнение

ГОТОВНОСТЬ

P_0

P_0	P_1	P_2	P_3
-------	-------	-------	-------

Алгоритмы планирования

SJF (Shortest Job First) приближение

$\tau(n)$ – величина n -го CPU burst

$T(n+1)$ – предсказание для $n+1$ -го CPU burst

α – параметр от 0 до 1

$$T(n+1) = \alpha \tau(n) + (1 - \alpha)T(n),$$

$T(0)$ – произвольно

Если $\alpha = 0$, то $T(n+1) = T(n) = \dots = T(0)$,
нет учета последнего поведения

Если $\alpha = 1$, то $T(n+1) = \tau(n)$,
нет учета предыстории

Алгоритмы планирования

Гарантированное планирование

В системе разделения времени N пользователей:

T_i – время нахождения i -го пользователя в системе

τ_i – суммарное процессорное время процессов i -го пользователя

$\tau_i \ll T_i / N$ – пользователь обделен

$\tau_i \gg T_i / N$ – пользователю благоволят

$(\tau_i N) / T_i$ – коэффициент справедливости.

На исполнение выбираются готовые процессы пользователя с наименьшим коэффициентом справедливости

Алгоритмы планирования

Приоритетное планирование

Каждому процессу процессор выделяется в соответствии с приписанным к нему числовым значением - приоритетом

Параметры для назначения приоритета бывают:

- внешние**
- внутренние**

Политика изменения приоритета:

- статический приоритет**
- динамический приоритет**

Алгоритмы планирования

Приоритетное планирование НЕВЫТЕСНЯЮЩИЙ

Процессы	P0	P1	P2	P3
Продолжительность CPU burst	6	2	5	5
Момент появления в очереди	0	2	6	0
Приоритет	4	3	2	1

время	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
P ₀	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И	И	И
P ₁			Г	Г	Г	И	И											
P ₂							Г	И	И	И	И	И						
P ₃	И	И	И	И	И													

исполнение

ГОТОВНОСТЬ

P₀

P ₀	P ₁	P ₂	P ₃
----------------	----------------	----------------	----------------

Алгоритмы планирования

Приоритетное планирование вытесняющий

Процессы	P0	P1	P2	P3
Продолжительность CPU burst	6	2	5	5
Момент появления в очереди	0	2	6	0
Приоритет	4	3	2	1

• время	•	1•	2•	3•	4•	5•	6•	7•	8•	9•	10•	11•	12•	13•	14•	15•	16•	17•	18•
• P ₀	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	Г	И	И	И	И	И	И	И	И
• P ₁	•	•	Г	Г	Г	И	Г	Г	Г	Г	Г	И	•	•	•	•			
• P ₂	•	•	•	•	•	•	И	И	И	И	И	•	•	•	•	•			
• P ₃	И	И	И	И	И	•	•	•	•	•	•	•	•	•	•	•			

исполнение

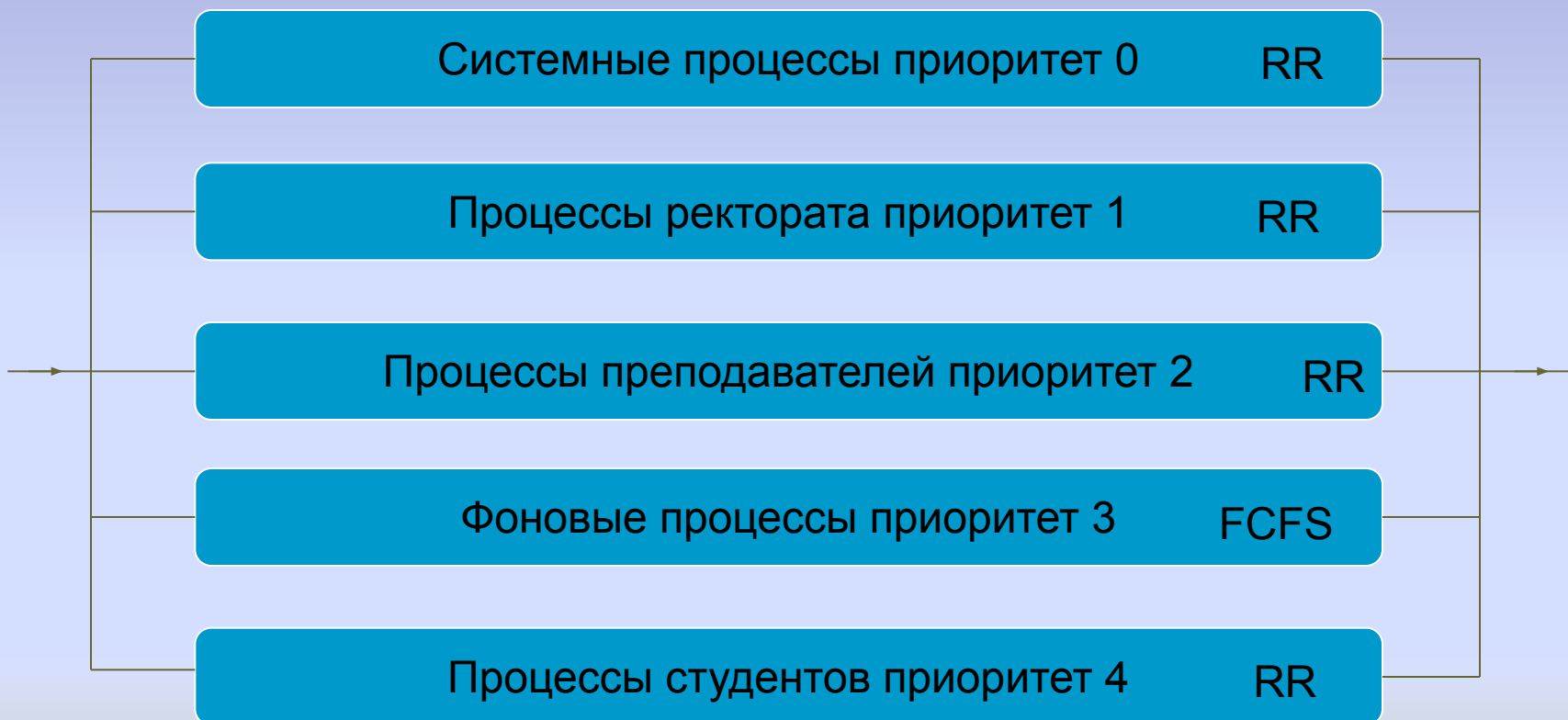
ГОТОВНОСТЬ

P₀

P ₀	P ₁	P ₂	P ₃
----------------	----------------	----------------	----------------

Алгоритмы планирования

Многоуровневые очереди (Multilevel Queue)



Алгоритмы планирования

Многоуровневые очереди с обратной связью (Multilevel Feedback Queue)



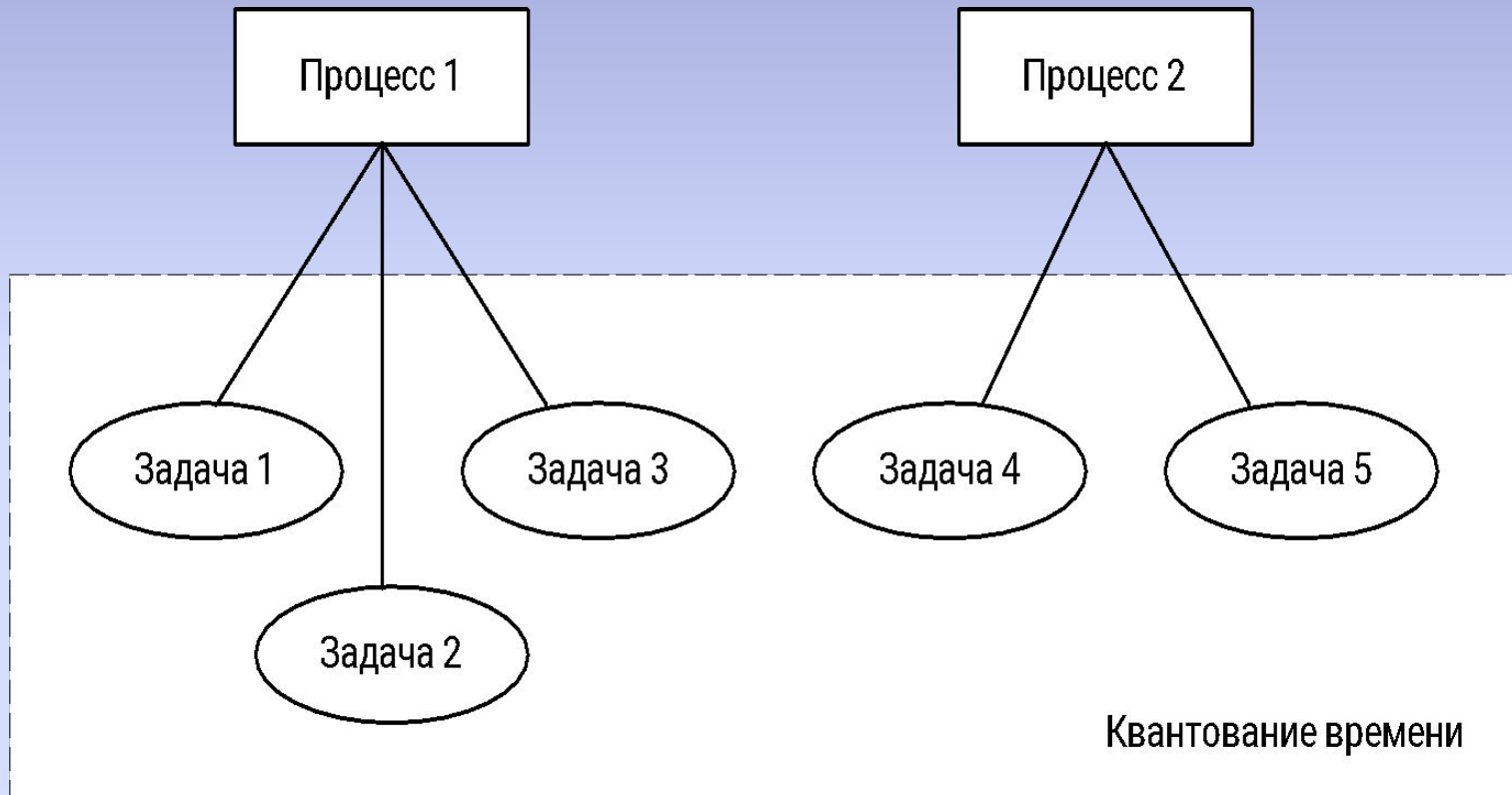
Алгоритмы планирования

Многоуровневые очереди с обратной связью (Multilevel Feedback Queue)

Для полного описания необходимо задать

- количество очередей в состоянии *готовность*
- алгоритм планирования между очередями
- алгоритмы планирования внутри очередей
- куда помещается родившийся процесс
- правила перевода процессов из одной очереди в другую

Квантование времени для задач



Планирование Windows NT

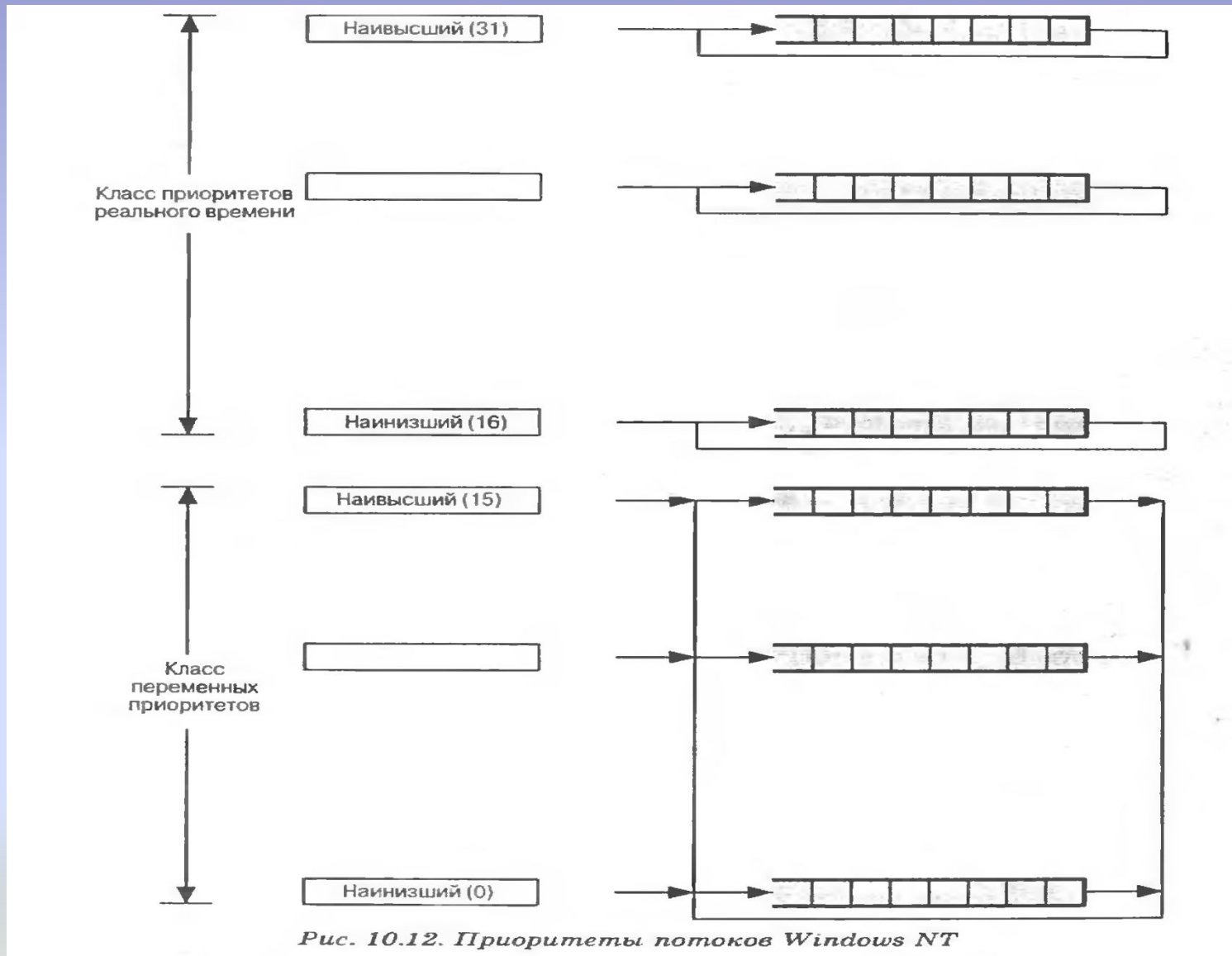


Рис. 10.12. Приоритеты потоков Windows NT

Приоритеты Windows NT

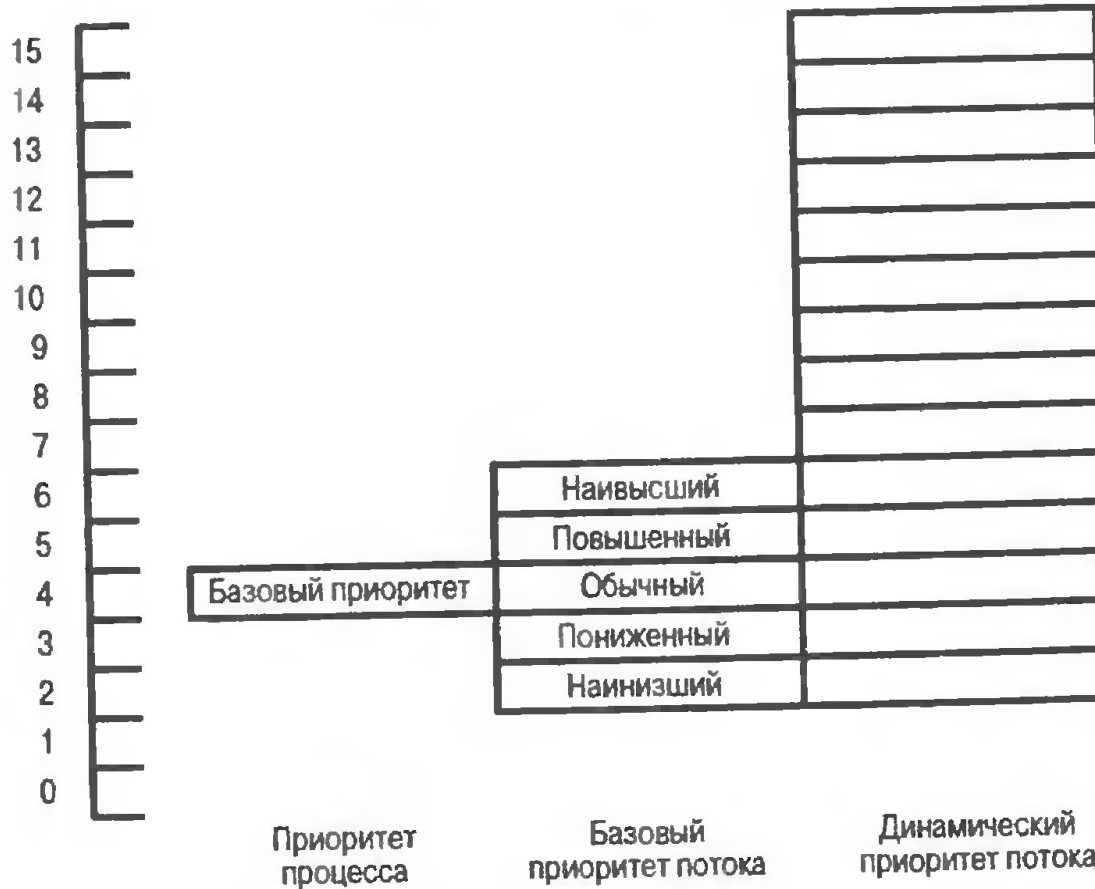


Рис. 10.13. Пример отношения приоритетов в Windows NT

Классы приоритета процессов

<i>Класс приоритета</i>	<i>Уровень приоритета</i>
REALTIME_PRIORITY_CLASS	24 - процессы реального времени
HIGH_PRIORITY_CLASS	13 - высокоприоритетные процессы
NORMAL_PRIORITY_CLASS	9 или 7 - обычные процессы
IDLE_PRIORITY_CLASS	4 - низкоприоритетные процессы

Относительный приоритет задач

<i>Значение</i>	<i>Относительное изменение уровня приоритета</i>
THREAD_PRIORITY_TIME_CRITICAL	Устанавливается абсолютный уровень приоритета 15 или 31
THREAD_PRIORITY_HIGHEST	+2
THREAD_PRIORITY_ABOVE_NORMAL	+1
THREAD_PRIORITY_NORMAL	0
THREAD_PRIORITY_BELOW_NORMAL	-1
THREAD_PRIORITY_LOWEST	-2
THREAD_PRIORITY_IDLE	Устанавливается абсолютный уровень приоритета 1 или 16

Функции Win32API для управления приоритетами задач и процессов

CreateProcess – создание процесса

```
BOOL CreateProcess(  
    LPCTSTR lpApplicationName, // указатель на имя исполняемого  
        // модуля  
    LPTSTR lpCommandLine, // указатель на командную строку  
    LPSECURITY_ATTRIBUTES lpProcessAttributes, // указатель на  
        // атрибуты защиты процесса  
    LPSECURITY_ATTRIBUTES lpThreadAttributes, // указатель на  
        // атрибуты защиты задачи  
    BOOL bInheritHandles, // флаг наследования идентификатора  
    DWORD dwCreationFlags, // флаги создания процесса  
    LPVOID lpEnvironment, // указатель на блок среды выполнения  
    LPCTSTR lpCurrentDirectory, // указатель на имя текущего  
        // каталога  
    LPSTARTUPINFO lpStartupInfo, // указатель на структуру  
        // STARTUPINFO  
    LPPROCESS_INFORMATION lpProcessInformation); // указатель на  
        // структуру PROCESS_INFORMATION
```

Функции Win32API для управления приоритетами задач и процессов

CreateThread – создание задачи (потока, цепочки)

```
HANDLE CreateThread(  
    LPSECURITY_ATTRIBUTES lpThreadAttributes, // атрибуты защиты  
    DWORD dwStackSize,    // начальный размер стека в байтах  
    LPTHREAD_START_ROUTINE lpStartAddress, // адрес функции  
                                // задачи  
    LPVOID lpParameter,    // параметры для задачи  
    DWORD dwCreationFlags, // параметры создания задачи  
    LPDWORD lpThreadId);   // адрес переменной для  
                            // идентификатора задачи
```

Функции Win32API для управления приоритетами задач и процессов

Управление запущенными задачами

```
BOOL SetThreadPriority(  
HANDLE hThread, // идентификатор задачи  
int nPriority); // новый уровень приоритета задачи  
  
int GetThreadPriority(HANDLE hThread);  
  
DWORD SuspendThread(HANDLE hThread);  
  
DWORD ResumeThread(HANDLE hThread);  
  
VOID Sleep(DWORD cMilliseconds); // время в миллисекундах  
  
VOID ExitThread(DWORD dwExitCode);  
  
BOOL TerminateThread(  
HANDLE hThread, // идентификатор завершаемой задачи  
DWORD dwExitCode); // код завершения
```

Традиционное планирование UNIX

$$CPU_j(i) = \frac{CPU_j(i-1)}{2}$$
$$P_j(i) = Base_j + \frac{CPU_j(i-1)}{2} + nice_j,$$

где

$CPU_j(i)$ — мера использования процессора процессом j на протяжении интервала i ;

$P_j(i)$ — приоритет процесса j в начале интервала i (меньшее значение соответствует большему приоритету);

$Base_j$ — базовый приоритет процесса j ;

$nice_j$ — указываемый пользователем коэффициент.

Традиционное планирование UNIX

Таблица 9.8. Пример традиционного планирования процессов в UNIX

Время	Процесс А		Процесс В		Процесс С	
	Приоритет	Счетчик	Приоритет	Счетчик	Приоритет	Счетчик
0	60	0 1 2 • • 60	60	0	60	0
1	75	30	60	0 1 2 • • 60	60	0
2	67	15	75	30	60	0 1 2 • • 60
3	63	7 8 9 • • 67	67	15	75	30
4	76	33	63	7 8 9 • • 67	67	15
5	68	16	76	33	63	7

Штриховка указывает выполняющийся процесс