Планирование процессов

Многозадачность

- ОС является *многозадачной*, если она способна чередовать выполнение нескольких процессов, создавая видимость, что в каждый момент времени работает более одного процесса
- *Кооперативная* многозадачность

Добровольное прерывание выполнения процесса им самим называется *уступкой*

- *Приоритетная* многозадачность

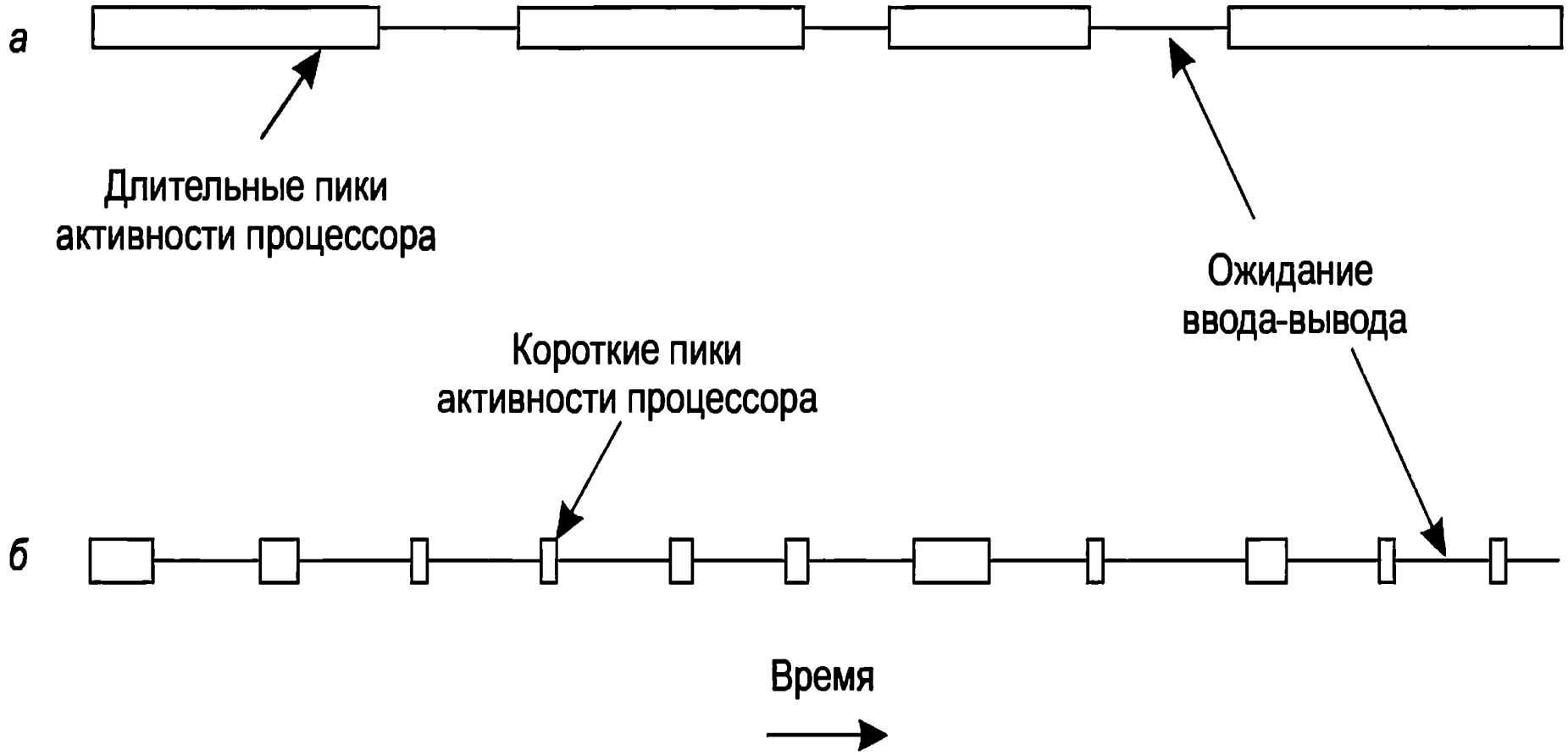
Квант времени процесса – время, в течение которого процесс может выполняться, прежде чем планировщик его прервет

Системный планировщик Linux

➢Компонент ядра, определяющий, какой из процессов должен выполняться, в какой именно момент времени и насколько долго

Версия ядра Linux	Тип планировщика
До версии 2.4	Простой планировщик, плохо поддающийся масштабированию
2.5	Планировщик типа *O*(1)
2.6	Планировщик типа RSDL (Rotating Staircase Deadline – Циклический ступенчатый граничный планировщик)
2.6.23	Планировщик типа CFS (Completely Fair Scheduler – Полностью справедливый планировщик)

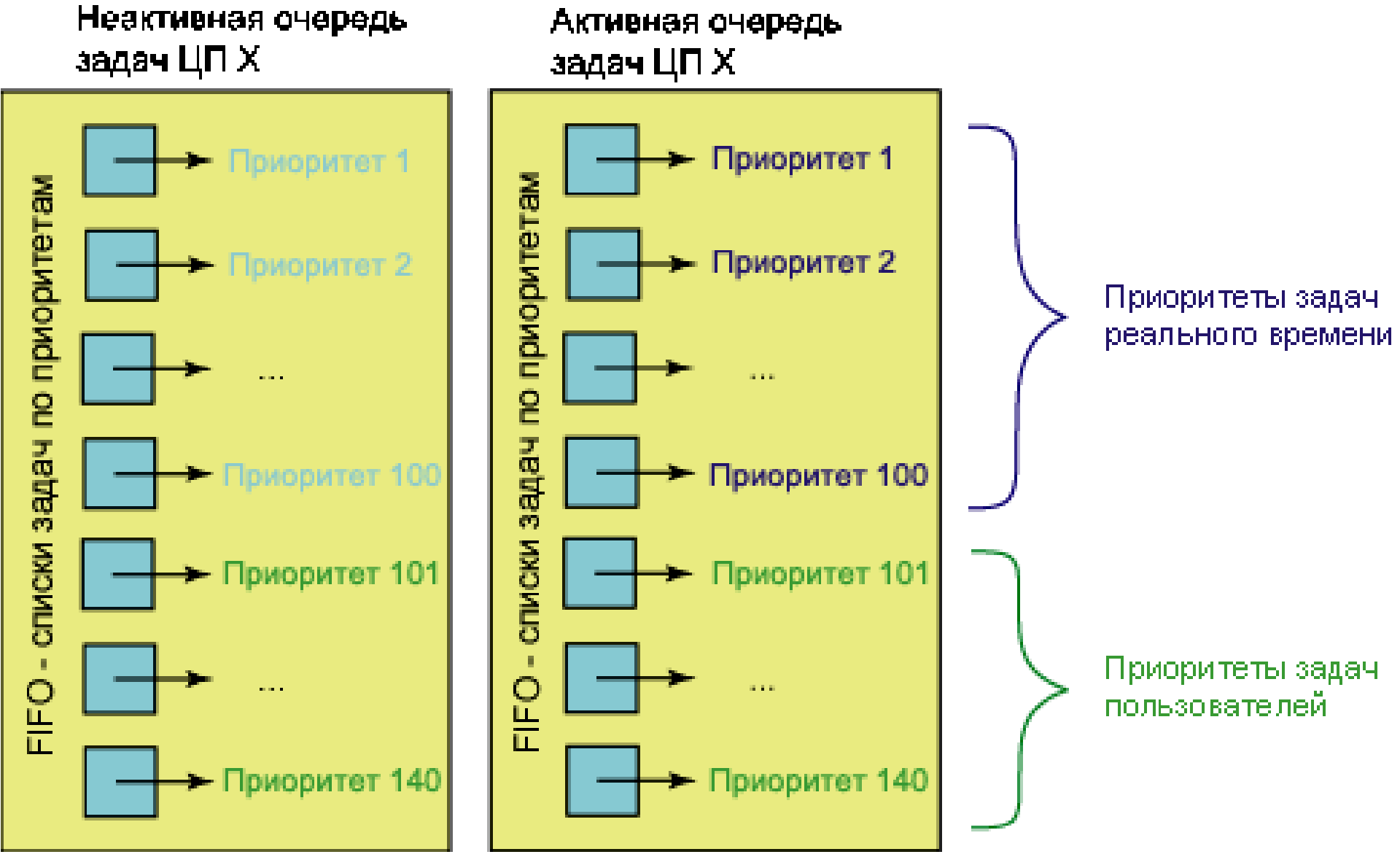
Поведение процесса



Приоритет процессов

- Цель приоритетного планирования состоит в упорядочении процессов в соответствии с их важностью и необходимостью использования процессорного времени
- Процессы с более высоким приоритетом должны выполняться раньше тех, которые имеют более низкий приоритет
- Процессы с одинаковым приоритетом планируются к выполнению по циклическому алгоритму (Round Robin)

Планирование процессов в Linux



Планирование в Linux. Приоритеты

Значение **`nice`**

-20 ... +19 (значение по умолчанию = 0)

Запуск команды с определённым приоритетом:

`nice -n значение_nice команда`

Изменение приоритета уже запущенного процесса:

`renice -n значение_nice id_процесса`

Планирование в Linux. Приоритеты

```
[ovm@ovm-pc ~]$ ps -el
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 30827 ep_pol ?        00:00:01 systemd
1 S     0     2     0  0  80   0 -     0 kthrea ?        00:00:00 kthreadd
1 S     0     3     2  0  80   0 -     0 smpboo ?        00:00:00 ksoftirqd/0
1 S     0     4     2  0  80   0 -     0 worker ?        00:00:00 kworker/0:0
1 S     0     5     2  0  60 -20 -     0 worker ?        00:00:00 kworker/0:0H
1 S     0     6     2  0  80   0 -     0 worker ?        00:00:00 kworker/u2:0
1 S     0     7     2  0  80   0 -     0 rcu_gp ?        00:00:00 rcu_sched
1 S     0     8     2  0  80   0 -     0 rcu_gp ?        00:00:00 rcu_bh
1 S     0     9     2  0  80   0 -     0 rcu_no ?        00:00:00 rcuos/0
1 S     0    10     2  0  80   0 -     0 rcu_no ?        00:00:00 rcuob/0
1 S     0    11     2  0 -40   - -     0 smpboo ?        00:00:00 migration/0
5 S     0    12     2  0 -40   - -     0 smpboo ?        00:00:00 watchdog/0
1 S     0    13     2  0  60 -20 -     0 rescue ?        00:00:00 khelper
5 S     0    14     2  0  80   0 -     0 devtmp ?        00:00:00 kdevtmpfs
1 S     0    15     2  0  60 -20 -     0 rescue ?        00:00:00 netns
1 S     0    16     2  0  60 -20 -     0 rescue ?        00:00:00 perf
1 S     0    17     2  0  60 -20 -     0 rescue ?        00:00:00 writeback
1 S     0    18     2  0  85   5 -     0 ksm_sc ?        00:00:00 ksmd
1 S     0    19     2  0  99  19 -     0 khugep ?        00:00:00 khugepaged
1 S     0    20     2  0  60 -20 -     0 rescue ?        00:00:00 crypto
1 S     0    21     2  0  60 -20 -     0 rescue ?        00:00:00 kintegrityd
1 S     0    22     2  0  60 -20 -     0 rescue ?        00:00:00 bioset
1 S     0    23     2  0  60 -20 -     0 rescue ?        00:00:00 kblockd
1 S     0    24     2  0  60 -20 -     0 rescue ?        00:00:00 ata_sff
1 S     0    25     2  0  60 -20 -     0 rescue ?        00:00:00 md
1 S     0    26     2  0  60 -20 -     0 rescue ?        00:00:00 devfreq_wq
1 S     0    27     2  0  80   0 -     0 worker ?        00:00:00 kworker/0:1
1 S     0    29     2  0  80   0 -     0 kswapd ?        00:00:00 kswapd0
1 S     0    30     2  0  80   0 -     0 fsnoti ?        00:00:00 fsnotify_mark
1 S     0    40     2  0  60 -20 -     0 rescue ?        00:00:00 kthrotld
1 S     0    41     2  0  60 -20 -     0 rescue ?        00:00:00 acpi_thermal_pm
1 S     0    42     2  0  80   0 -     0 worker ?        00:00:00 kworker/u2:1
1 S     0    43     2  0  80   0 -     0 scsi_e ?        00:00:00 scsi_eh_0
1 S     0    44     2  0  60 -20 -     0 rescue ?        00:00:00 scsi_tmf_0
1 S     0    45     2  0  80   0 -     0 scsi_e ?        00:00:00 scsi_eh_1
1 S     0    46     2  0  60 -20 -     0 rescue ?        00:00:00 scsi_tmf_1
1 S     0    47     2  0  80   0 -     0 scsi_e ?        00:00:00 scsi_eh_2
1 S     0    48     2  0  60 -20 -     0 rescue ?        00:00:00 scsi_tmf_2
1 S     0    49     2  0  80   0 -     0 worker ?        00:00:00 kworker/u2:2
1 S     0    50     2  0  80   0 -     0 worker ?        00:00:00 kworker/u2:3
1 S     0    51     2  0  60 -20 -     0 rescue ?        00:00:00 kpsmoused
1 S     0    52     2  0  60 -20 -     0 rescue ?        00:00:00 dm_bufio_cache
```

Планирование в Linux. Приоритеты

$$\text{базовый квант времени (в мс)} = \begin{cases} (140 - \text{статический приоритет}) \times 20, \\ \text{если статический приоритет} < 120 \\ (140 - \text{статический приоритет}) \times 5, \\ \text{если статический приоритет} \geq 120 \end{cases}$$

Описание	Статический приоритет	Значение nice	Базовый квант времени	Дельта интерактивности	Порог времени сна
Наивысший статический приоритет	100	–20	800 мс	–3	299 мс
Высокий статический приоритет	110	–10	600 мс	–1	499 мс
Статический приоритет по умолчанию	120	0	100 мс	+2	799 мс
Низкий статический приоритет	130	+10	50 мс	+4	999 мс
Самый низкий статический приоритет	139	+19	5 мс	+6	1199 мс

Динамическое назначение приоритетов

- Приоритет процессов, ограниченных скоростью ввода-вывода, в качестве награды, снижается на величину до пяти уровней
- Процессы, ограниченные производительностью ЦП, «штрафуются» повышением приоритета также на величину до пяти уровней
- Принадлежность процесса к классу ограниченных скоростью ввода-вывода или ограниченных производительностью ЦП определяется с помощью эвристической процедуры вычисления *интерактивности*
- Коррекция приоритета производится только для пользовательских процессов

Динамический приоритет

$$\text{динамический приоритет} = \max(100, \min(\text{статический приоритет} - \text{бонус} + 5, 139))$$

Процесс считается интерактивным, если он удовлетворяет следующему соотношению:

$$\text{бонус} - 5 \geq \underbrace{\text{статический приоритет} / 4 - 28}_{\text{дельта интерактивности}}$$

Планирование в Linux. Приоритеты

Приоритеты реального времени

0 … 99

- Бóльшим значениям соответствует и бóльший приоритет

- Все процессы реального времени имеют более высокий приоритет по сравнению с обычными процессами

Планирование в Linux. Приоритеты

```
[ovm@ovm-pc ~]$ ps -eo state,uid,pid,ppid,rtprio,time,comm
S   UID   PID  PPID RTPRIO     TIME COMMAND
S     0     1     0      -  00:00:01 systemd
S     0     2     0      -  00:00:00 kthreadd
S     0     3     2      -  00:00:01 ksoftirqd/0
S     0     5     2      -  00:00:00 kworker/0:0H
S     0     7     2      -  00:00:00 rcu_sched
S     0     8     2      -  00:00:00 rcu_bh
S     0     9     2      -  00:00:00 rcuos/0
S     0    10     2      -  00:00:00 rcuob/0
S     0    11     2     99  00:00:00 migration/0
S     0    12     2     99  00:00:00 watchdog/0
S     0    13     2      -  00:00:00 khelper
S     0    14     2      -  00:00:00 kdevtmpfs
S     0    15     2      -  00:00:00 netns
S     0    16     2      -  00:00:00 perf
S     0    17     2      -  00:00:00 writeback
S     0    18     2      -  00:00:00 ksmd
S     0    19     2      -  00:00:00 khugepaged
S     0    20     2      -  00:00:00 crypto
S     0    21     2      -  00:00:00 kintegrityd
S     0    22     2      -  00:00:00 bioset
S     0    23     2      -  00:00:00 kblockd
S     0    24     2      -  00:00:00 ata_sff
S     0    25     2      -  00:00:00 md
S     0    26     2      -  00:00:00 devfreq_wq
S     0    29     2      -  00:00:00 kswapd0
S     0    30     2      -  00:00:00 fsnotify_mark
S     0    40     2      -  00:00:00 kthrotld
S     0    41     2      -  00:00:00 acpi_thermal_pm
S     0    43     2      -  00:00:00 scsi_eh_0
S     0    44     2      -  00:00:00 scsi_tmf_0
S     0    45     2      -  00:00:00 scsi_eh_1
S     0    46     2      -  00:00:00 scsi_tmf_1
S     0    47     2      -  00:00:00 scsi_eh_2
S     0    48     2      -  00:00:00 scsi_tmf_2
S     0    51     2      -  00:00:00 kpsmoused
S     0    52     2      -  00:00:00 dm_bufio_cache
S     0    53     2      -  00:00:00 ipv6_addrconf
S     0    55     2      -  00:00:00 deferwq
S     0    56     2      -  00:00:00 kworker/u2:4
```

Планирование в Linux. Квант времени

- Квант времени – числовое значение, определяющее, как долго может выполняться процесс до того момента, пока он не будет вытеснен
- Слишком большое значение кванта времени приведет к ухудшению интерактивной производительности системы
- Слишком малое значение кванта времени приведет к возрастанию накладных расходов на переключение между процессами
- В планировщике Linux процессам выделяется *доля* (portion) процессорного времени
- На величину доли влияет загруженность системы и значение параметра **`nice`**

Планирование в Linux. Классы планировщика

- В планировщике одновременно сосуществуют несколько разных встроенных алгоритмов (*классы планировщика*), предназначенных для планирования процессов только определенного типа
- Каждый класс планировщика имеет свой приоритет
- Решение о том, какой процесс будет запущен следующим принимает класс планировщика, имеющий наивысший приоритет и соответствующий типу готового процесса

Планировщик CFS (Completely Fair Scheduler – Полностью справедливый планировщик)

- Класс планировщика для обычных процессов (**`SCHED_NORMAL`**)

Стратегии планирования в режиме реального времени

- Классы планирования для процессов реального времени: **`SCHED_FIFO`** и **`SCHED_RR`**
- Эти классы реализуются не в планировщике CFS, а в специальном планировщике реального времени
- **`SCHED_FIFO`** – простой алгоритм планирования по принципу «Первым пришел – первым обслужен» без использования квантов времени
- **`SCHED_RR`** – это **`SCHED_FIFO`** с квантованием, циклический (round-robin) алгоритм планирования

Использованная литература

1. Планировщик задач Linux http://www.ibm.com/developerworks/ru/library/l-scheduler/
2. Лав Р. Ядро Linux: описание процесса разработки, 3-е изд. – М.: Вильямс, 2015. – 496 с. (глава 4)