Управление процессами и потоками

Алгоритмы планирования

Уровни планирования

- Долгосрочное
  – планирование загрузки новых процессов (минуты, десятки минут)
- Среднесрочное
  – выгрузка выполняющихся процессов на диск (swapping)
- Короткосрочное
  – планирование использования процессора (не более 100 ms)

Планирование потоков (процессов)

- Определение момента смены выполняемого процесса
- Выбор процесса на выполнение из очереди готовых процессов
- Переключение контекстов «старого» и «нового» процессов

Цели планирования

- Справедливость
- Эффективность
- Сокращение полного времени выполнения
- Сокращение времени ожидания
- Сокращение времени отклика
- …

Требования к алгоритмам

- Предсказуемость
- Минимальные накладные расходы
- Равномерная загрузка вычислительной системы
- Масштабируемость
- …

Параметры планирования

Статические

- Пользователь
- Приоритет
- Требования к времени использования процессора и ввода-вывода
- Требования к другим ресурсам

Динамические

- Время с момента выгрузки/загрузки процесса
- Использованное время CPU/ввода-вывода
- Используемая память
- Последние и ожидаемые CPU Burst и IO Burst

Распределение времени выполнения потока

A=1
B=2
Read c

} CPU Burst

Ожидание ввода

} I/O Burst

A=A+C*B
Print A

} CPU Burst

Ожидание вывода

} I/O Burst

Вытесняющее и невытесняющее планирование

Когда может производится планирование:

- поток переходит из состояния «исполнение» в состояние «закончил работу»
- поток переходит из состояния «исполнение» в состояние «ожидание»
- поток переходит из состояния «исполнение» в состояние «готовность» (например по прерыванию)
- поток переходит из состояния «ожидание» в состояние «готовность»

Невытесняющее планирование

Вытесняющее планирование

Алгоритм планирования First-Come, First-Served (FCFS)

- Невытесняющее планирование
- При переходе в состояние «готовность» ссылка на поток помещается в очередь (FIFO)
- Выбор нового потока на выполнение выполняется из начала очереди с удалением ссылки.

Алгоритм планирования First-Come, First-Served (FCFS)

Поток	P0	P1	P2
Продолжительность CPU Burst	13	4	1

Выполнение в порядке P0, P1, P2:

P0	и	и	и	и	и	и	и	и	и	и	и	и	и					
P1	г	г	г	г	г	г	г	г	г	г	г	г	г	и	и	и	и	
P2	г	г	г	г	г	г	г	г	г	г	г	г	г	г	г	г	г	и
T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Алгоритм планирования First-Come, First-Served (FCFS)

Выполнение в порядке P0, P1, P2:

P0	и	и	и	и	и	и	и	и	и	и	и	и	и					
P1	г	г	г	г	г	г	г	г	г	г	г	г	г	и	и	и	и	
P2	г	г	г	г	г	г	г	г	г	г	г	г	г	г	г	г	г	и
T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

	Время ожидания	Полное время выполнения
P0	0	13
P1	13	17
P2	17	18
Среднее	10	16

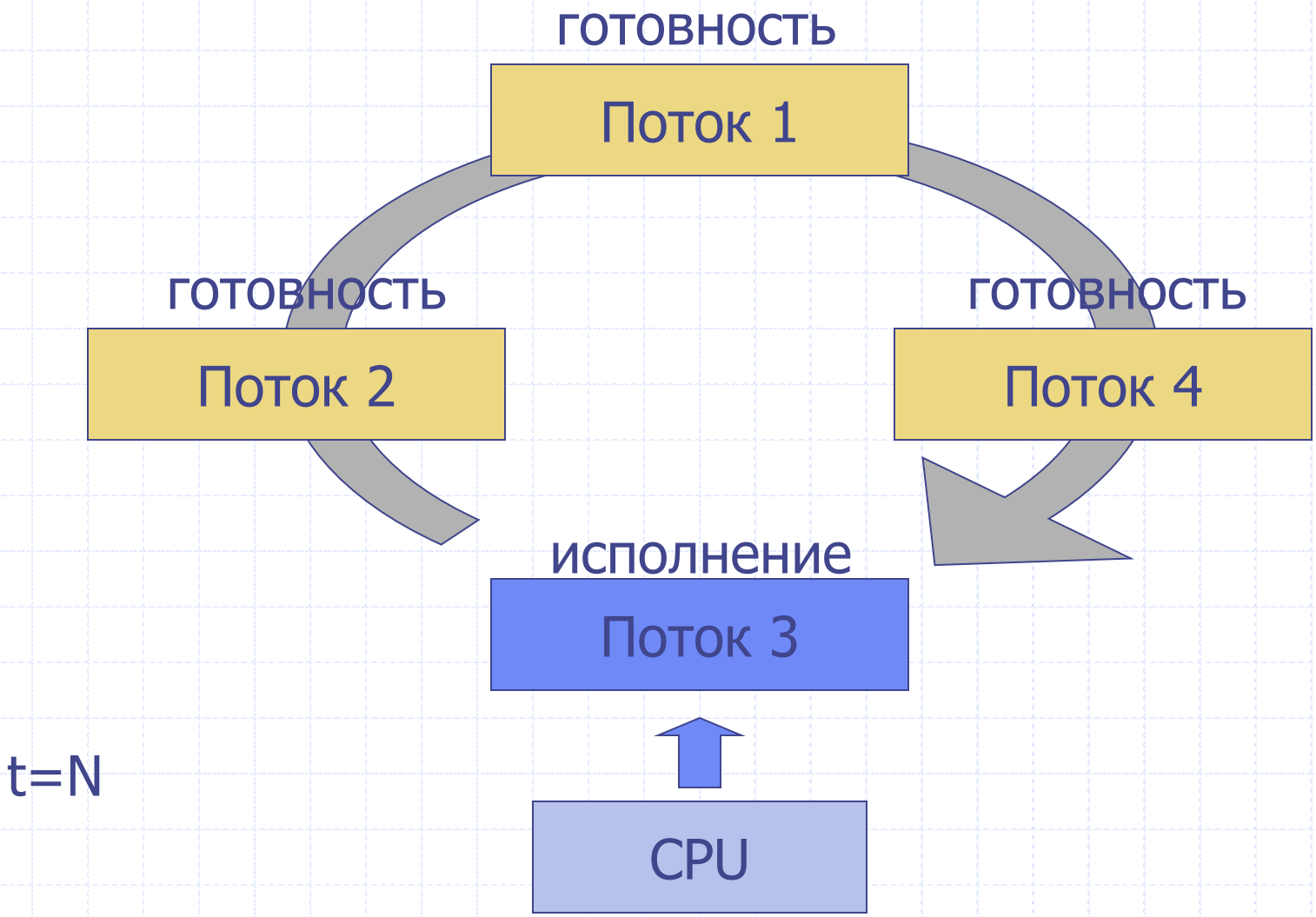
Алгоритм планирования First-Come, First-Served (FCFS)

Выполнение в порядке P2, P1, P0:

P2	и																
P1	г	и	и	и	и												
P0	г	г	г	г	г	и	и	и	и	и	и	и	и	и	и	и	и	и
T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

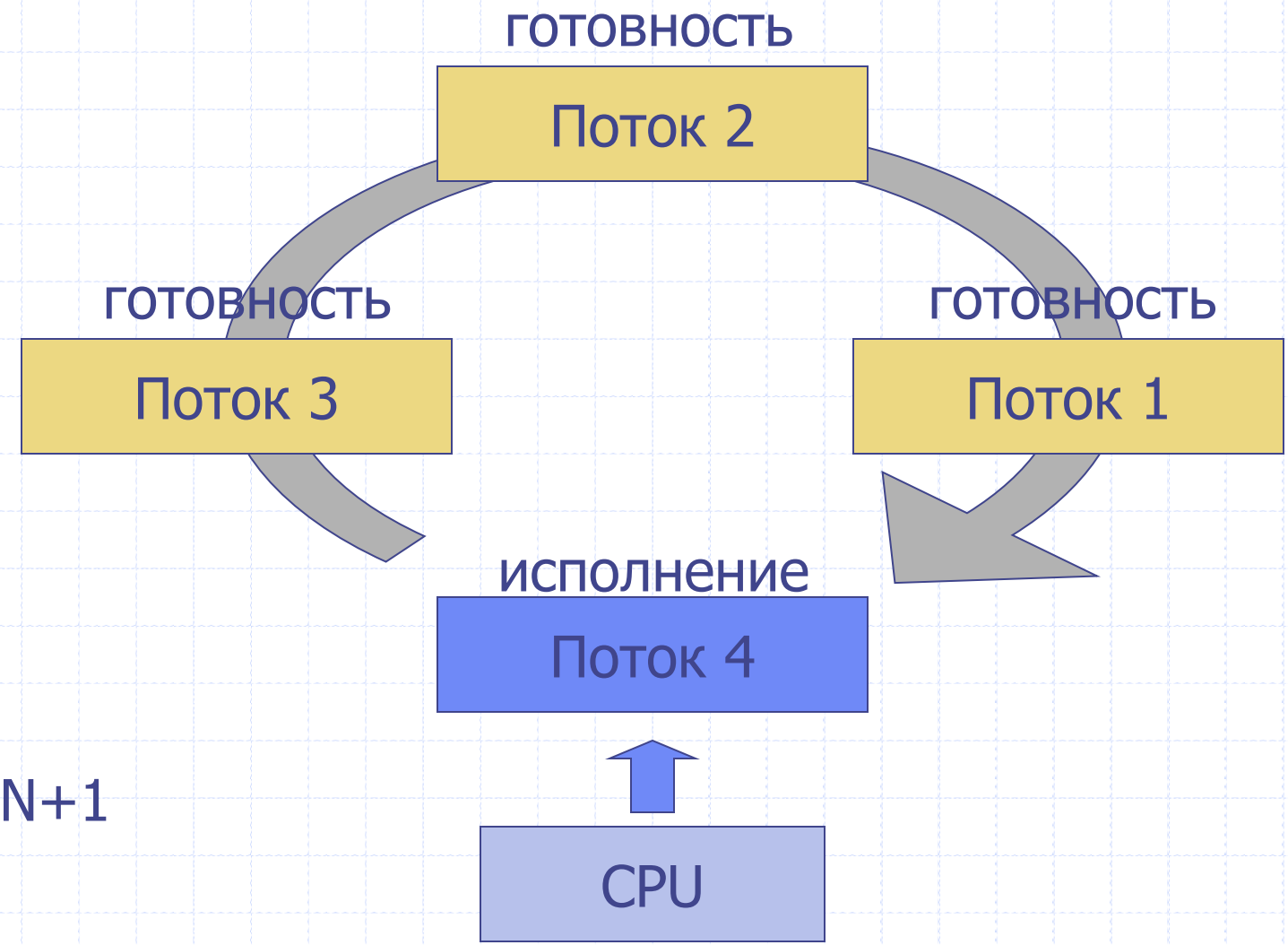
	Время ожидания	Полное время выполнения
P0	5	18
P1	1	5
P2	0	1
Среднее	2	8

Алгоритм планирования Round Robin (RR)



t=N

Алгоритм планирования Round Robin (RR)



t=N+1

Алгоритм планирования Round Robin (RR)

Вытесняющее планирование.

При переводе потока в состояние «исполнение» ему выделяется квант времени.

Если остаток текущего CPU burst меньше кванта времени, то поток сам возвращает управление ОС.
Иначе ОС переводит поток в состояние «готовность» по завершению кванта времени (по прерыванию).

Алгоритм планирования Round Robin (RR)

Выполнение с квантом времени = 4

P0	и	и	и	и	г	г	г	г	г	и	и	и	и	и	и	и	и	и
P1	г	г	г	г	и	и	и	и										
P2	г	г	г	г	г	г	г	г	и									
T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

	Время ожидания	Полное время выполнения
P0	5	18
P1	4	8
P2	8	9
Среднее	5.666666	11.666666

Алгоритм планирования Round Robin (RR)

Выполнение с квантом времени = 1

P0	и	г	г	и	г	и	г	и	г	и	и	и	и	и	и	и	и
P1	г	и	г	г	и	г	и	г	и								
P2	г	г	и														
T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

	Время ожидания	Полное время выполнения
P0	5	18
P1	5	9
P2	2	3
Среднее	4	10

Алгоритм планирования Shortest Job First (SJF)

Целесообразно выбирать для выполнения поток с минимальным значением CPU Burst.

$$
\begin{aligned}
&T(n+1) = \alpha t(n) + (1-\alpha)T(n) \\
&= \alpha t(n) + (1-\alpha)\big(\alpha t(n-1) + (1-\alpha)T(n-1)\big) \\
&= \alpha t(n) + \alpha(1-\alpha)t(n-1) + (1-\alpha)^2 T(n-1) \\
&= \alpha t(n) + \alpha(1-\alpha)t(n-1) + \alpha(1-\alpha)^2 t(n-2) + (1-\alpha)^3 T(n-2) \\
&= \alpha t(n) + \alpha(1-\alpha)t(n-1) + \alpha(1-\alpha)^2 t(n-2) + \cdots \\
&\qquad \cdots + \alpha(1-\alpha)^n t(0) + (1-\alpha)^{n+1} T(0)
\end{aligned}
$$

$$T(n+1) = \alpha t(n) + (1-\alpha)T(n)$$

$T(n)$ – прогноз на момент времени n, $T(0) = const$

$t(n)$ – реальная длительность на момент времени n

α – константа из $[0, \ldots, 1]$

Приоритетное планирование

- внутренние параметры
- вытесняющее

  При переходе в состояние «готовность» более приоритетный поток вытесняет менее приоритетный
- статическое

Приоритет постоянен

- внешние параметры
- не вытесняющее

  Приоритет влияет лишь на порядок выбора потоков на исполнение из очереди
- динамическое

Приоритет может меняться во время работы

Приоритеты процессов и потоков в Windows

- Потоки планируются на основании приоритета планирования
- Минимальный приоритет планирования – 0, максимальный – 31
- Приоритет 0 имеет поток обнуления неиспользуемых страниц памяти
- Потоки с одинаковым приоритетом рассматриваются как равные и планируются алгоритмом RoundRobin, квант 20 ms.
- Поток с более высоким приоритетом вытесняет менее приоритетный, даже если тот не доработал свой квант

Базовый приоритет потока

		Класс приоритета процесса					
		Idle	Below normal	Normal	Above normal	High	Realtime
Уровень приоритета потока	Idle	1	1	1	1	1	16
	Lowest	2	4	6	8	11	22
	Below normal	3	5	7	9	12	23
	Normal	4	6	8	10	13	24
	Above normal	5	7	9	11	14	25
	Highest	6	8	10	12	15	26
	Time critical	15	15	15	15	15	31

Динамический приоритет потока

- Планирование осуществляется на основе динамического приоритета потока
- Изначально динамический приоритет равен базовому
- Динамический приоритет повышается когда:
  - Активируется окно процесса с приоритетом Normal
  - Приоритет потока повышается при получении им пользовательского ввода
  - Выполняется условие ожидания потока
- Динамическое повышение не применяется для потоков с базовым приоритетом 16…32
- После повышения динамический приоритет уменьшается на 1 после каждого кванта
- Динамический приоритет не может быть меньше базового

Инверсия приоритета

Поток 1 Низкий приоритет	**Поток 2** Высокий приоритет	**Поток 3** Средний приоритет
Захватывает критическую секцию A	Ждёт	
Вытеснен потоком 2	Работает Пытается захватить A	
	Ждёт освобождения A	
Вытеснен потоком 3		Работает сколько хочет
Освобождает A		

Для выхода из этой ситуации планировщик случайным образом повышает приоритеты потоков, находящихся в состоянии готовности, что может позволить таким потокам завершить обработку критической секции.

Многоуровневые очереди

- Потоки разбиваются на группы
- У каждой группы есть своя очередь процессов в состоянии «готовность»
- Каждая очередь имеет фиксированный приоритет
- Внутри очередей могут применяться разные алгоритмы планирования

Многоуровневые очереди

Системные потоки
приоритет 0, RR

Потоки администраторов
приоритет 1, RR

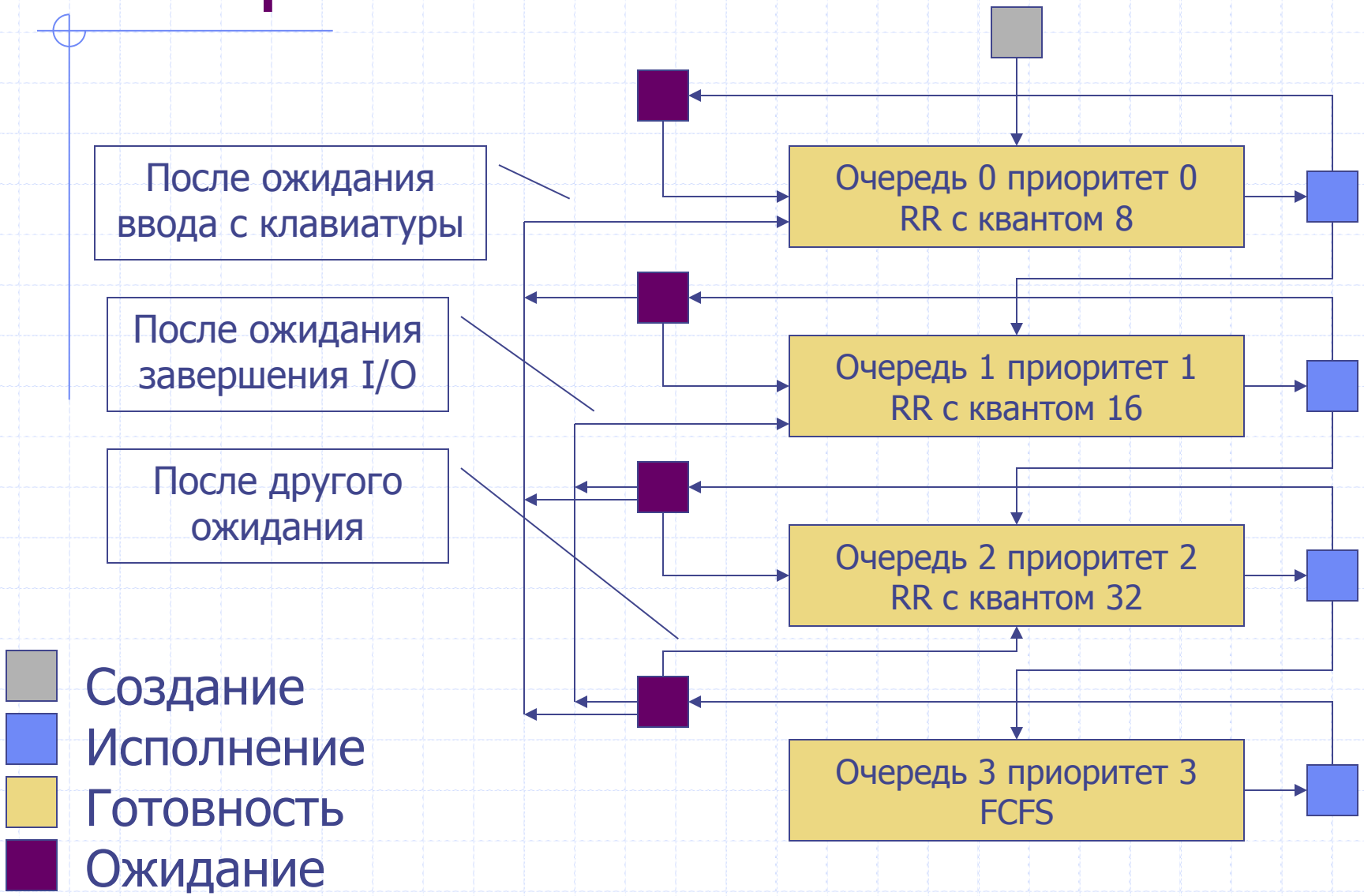
Потоки пользователей
приоритет 2, RR

Фоновые потоки
приоритет 3, FCFS

Многоуровневые очереди с обратной связью

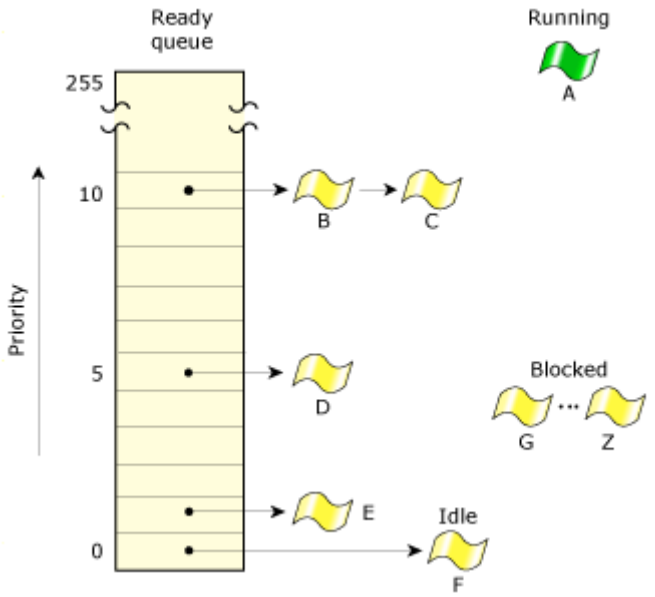
- Потоки могут перемещаться между очередями в зависимости от своего поведения (например, продолжительности последних CPU burst)

Многоуровневые очереди с обратной связью



Приоритеты потоков QNX

- Каждый поток имеет приоритет от 0 до 255
- Приоритет 0 – специальный системный поток
- Приоритеты 1…63 – пользовательские потоки
- Приоритет 64…255 – потоки root’a

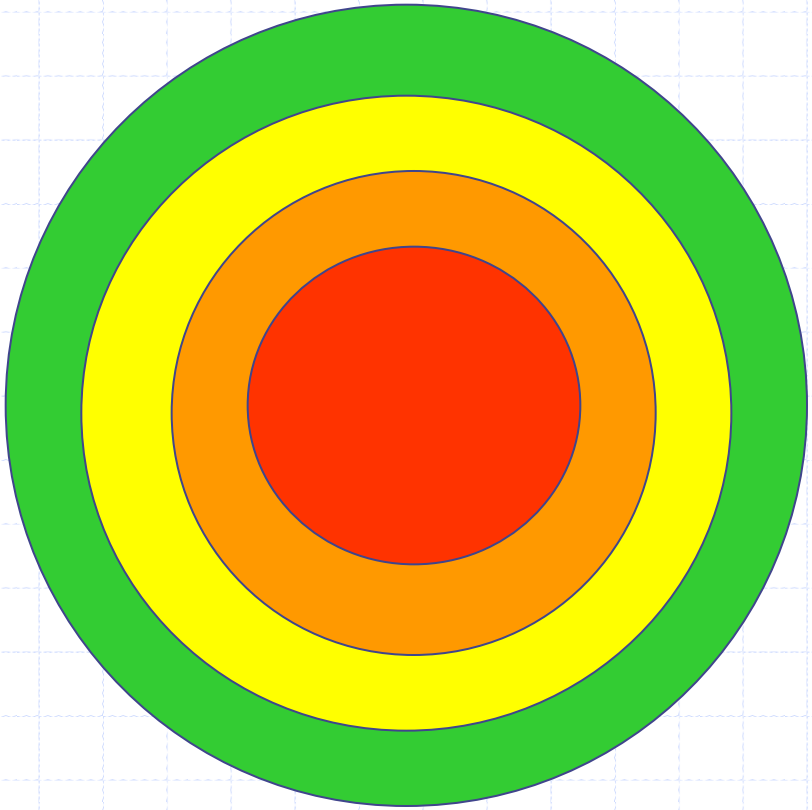


Алгоритмы планирования QNX

- Каждый поток может управляться по одному из алгоритмов:
  - FIFO
  - Round-Robin
  - Sporadic
- Алгоритмы работают только между потоками одного приоритета. Более приоритетный поток всегда вытесняет менее приоритетный.

Кольца защиты

Иерархические домены защиты называют **кольцами защиты**



Кольцо 0 - ядро

Кольцо 1 - драйвера

Кольцо 2 - драйвера

Кольцо 3 - приложения

Режимы работы CPU

- Современные процессоры поддерживают режимы работы с разным уровнем привилегий (не менее двух)
- В привилегированном режиме возможно выполнение любых команд
- В режимах с меньшими уровнями часть команд может быть не доступна
  - Команды ввода-вывода
  - Обращения к определённым областям памяти
  - …
- ОС могут использовать режимы CPU для реализации колец защиты

Обращения процессов к операционной системе

Существуют два способа организации взаимодействия пользовательского процесса и ОС:

- Системные вызовы
- Передача сообщений

Системные вызовы

- Для реализации системных вызовов создаётся библиотека, содержащая соответствующие функции
- Процесс обращается к функции
- Функция подготавливает данные и передаёт управление коду защищённого режима
  - Прерывания
  - Специальная команда процессора
- Код ядра анализирует параметры функции, проверяет допустимость операции и выполняет запрошенную функцию.

Передача сообщений

- Пользовательский процесс создаёт сообщение, описывающее требуемое действие
- С помощью специальной функции сообщение передаётся процессу ОС
- Пользовательский процесс блокируется на время выполнения запроса
- По завершению обработки ОС отсылает сообщение с результатами пользовательскому процессу и тот разблокируется