Операционные системы

Управление центральным процессором и объединение ресурсов

Управление центральным процессором…

Общие аспекты

Общие аспекты

- Контекст процесса, переключение контекста
- Дисциплины обслуживания
- Модели многопоточности

Контекст процесса

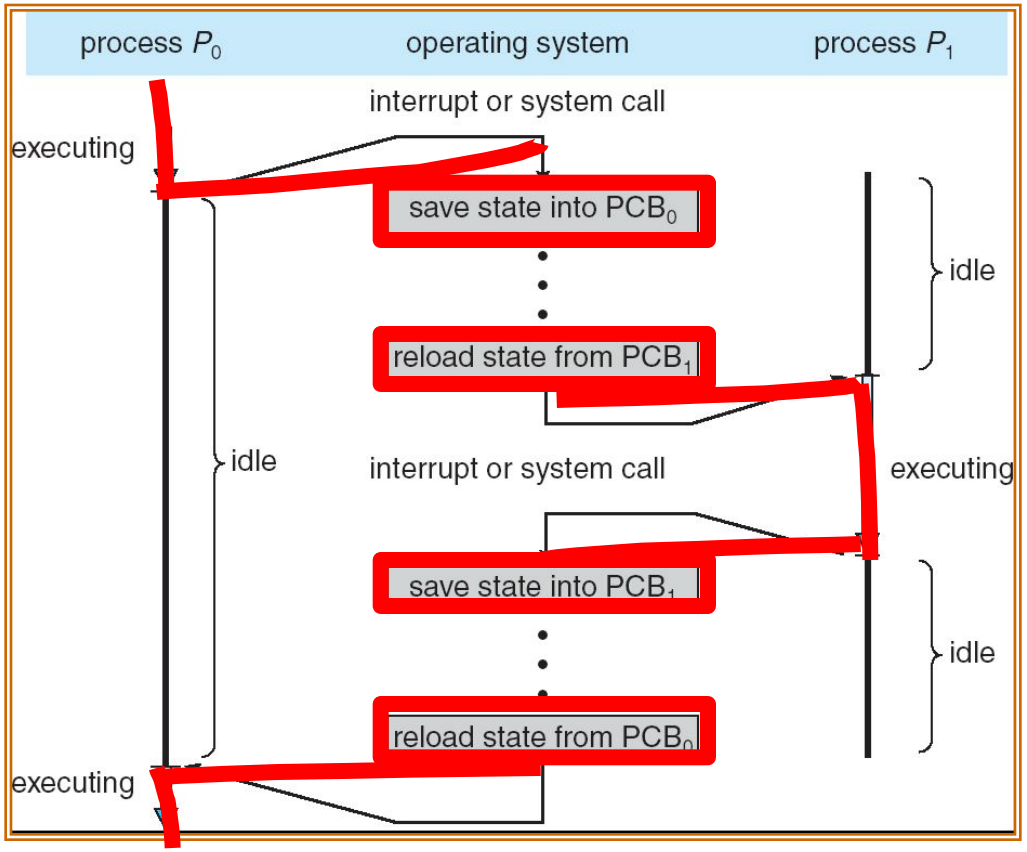
- Контекст – хранит состояние регистров, состояние программного счетчика, режим работы процессора, незавершенные операции ввода-вывода, информация о выполненных системных вызовах.
- Для хранения информации, необходимой для совершения операции над процессом, используется структура данных **PCB** (Process Control Block – блок управления процессом).
- Конкретный состав PCB зависит ОС и обычно содержит:
  - регистровый контекст процессора – все регистры и программный счетчик;
  - системный контекст процессора (контекст ядра ОС).
- Кроме того к контексту процесса относят пользовательский контекст процесса, который включает код и данные, находящиеся в адресном пространстве процесса.

Process Control Block

- Регистровый контекст процесса:
  - программный счетчик (адрес команды, которая должна быть выполнена для него следующей);
  - содержимое регистров.
- Системный контекст процессора:
  - состояние процесса;
  - данные для планирования использования процессора и управления памятью;
  - учетная информация;
  - сведения об устройствах ввода-вывода, связанных с процессом.

Переключение контекста

- Переключение процессора с выполнения команд одного потока на выполнение команд другого называют переключением контекста.



Вопрос

- Как Вы думаете какое влияние оказывает частое переключение контекста на производительность системы?
- Почему?

Переключение контекста и производительность системы

- Время, затраченное на переключение контекста, не используется вычислительной системой для совершения полезной работы и представляет собой накладные расходы, снижающие производительность системы. Оно меняется от машины к машине и обычно находится в диапазоне от 1 до 1000 микросекунд.
- Чем чаще происходит переключение контекста, тем система более реактивна, но при этом обладает меньшей пропускной способностью, т.к. чаще происходит переключение контекста.

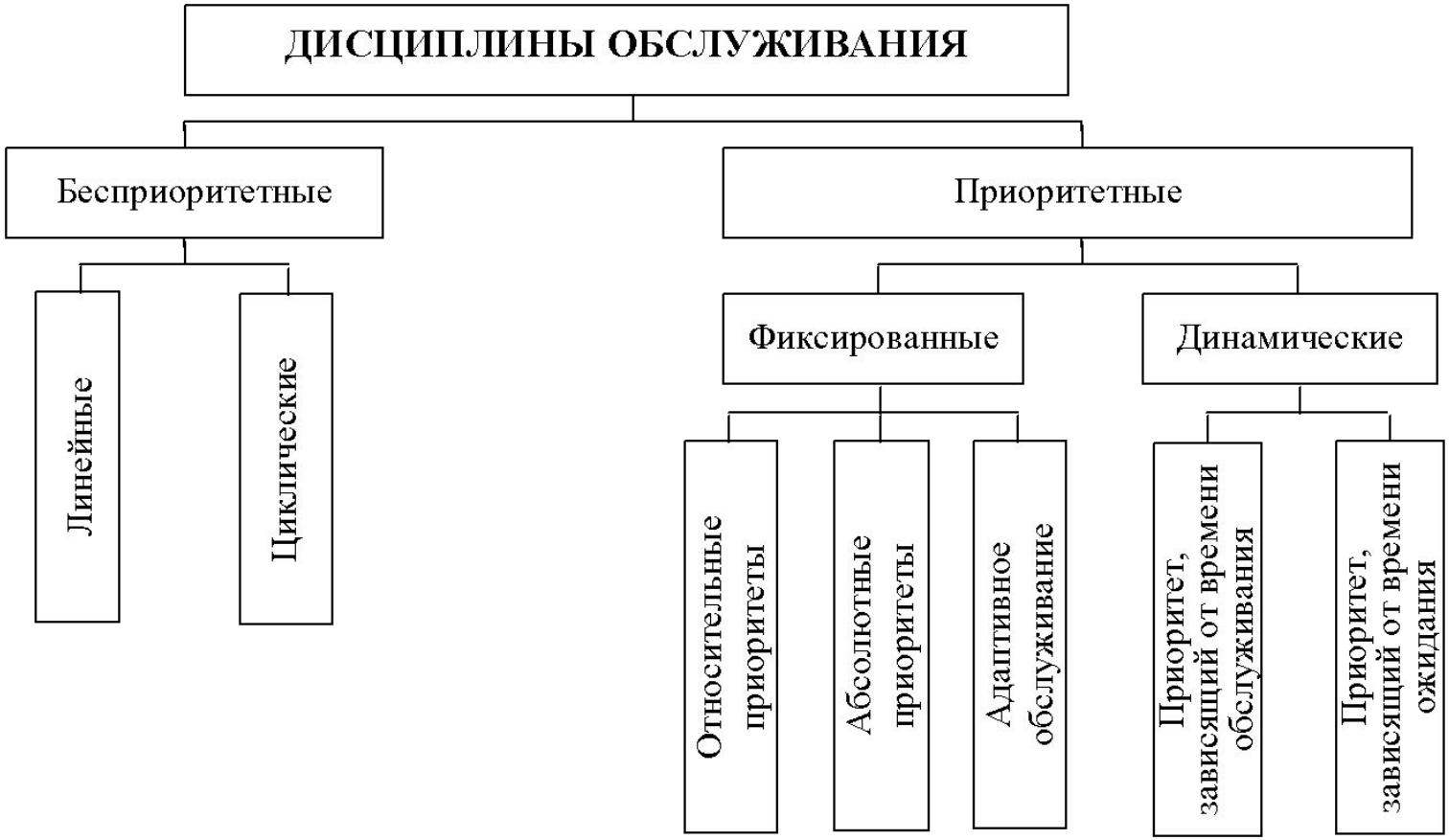
Сопутствующие факторы, влияющие на производительность

- При переключении контекста происходят аппаратные действия, влияющие на производительность:
  - очистка конвейера команд и данных процессора;
  - очистка TLB, отвечающего за страничное отображение линейных адресов на физические.
- Кроме того, следует учесть следующие факты, влияющие на состояние системы:
  - содержимое кэша (особенно кэша 1-го уровня) накопленное и «оптимизированное» под выполнение одного потока оказывается совершенно неприменимым к новому потоку, на который происходит переключение;
  - при переключении контекста, на поток, который до этого долгое время не использовался, многие страницы могут физически отсутствовать в оперативной памяти, что порождает подгрузку вытесненных страниц из вторичной памяти.

Методы снижения ресурсоемкости переключения контекста

- Использование многопоточности
  - при переключении контекста между потоками одного процесса, регистр CR3 не меняется и содержимое TLB сохраняется
- Размещение ядра ОС в адресном пространстве пользовательского процесса
  - при переключении контекста между user-space и kernel-space (и обратно), что, например, происходит при выполнении системных вызовов, регистр CR3 не меняется и содержимое TLB сохраняется
- Минимизация перемещения потоков при диспетчеризации в SMP-системе
  - улучшается эффективность работы кэша 2-го уровня
- Оптимизация восстановления контекста потока под операции с регистрами общего назначения
  - реальное сохранение/восстановление контекста регистров сопроцессора плавающей точки и MMX/SSE контекст происходит при первом обращении нового потока

Классификация дисциплин обслуживания (1)



Классификация дисциплин обслуживания (2)

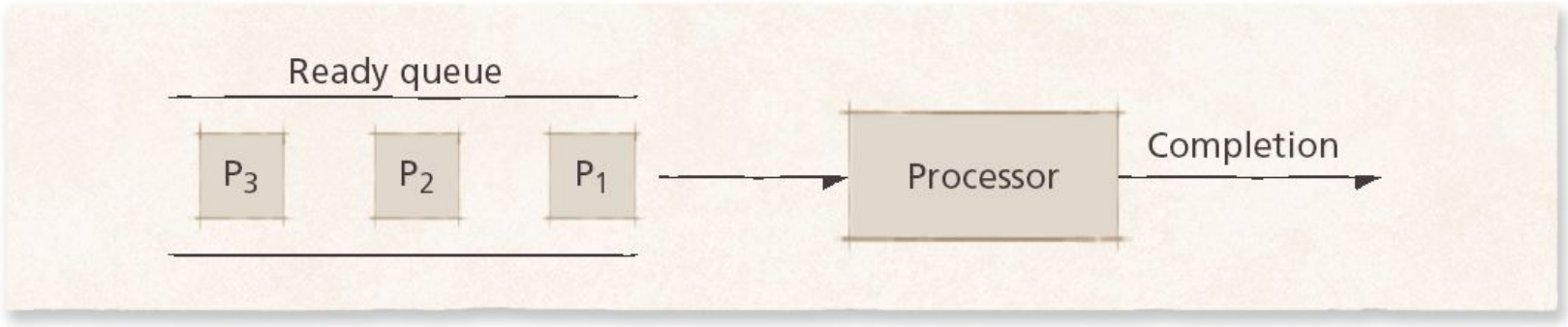
- *Бесприоритетные дисциплины* – выбор из очереди производится без учета относительной важности задач и времени их обслуживания.
- *Приоритетное обслуживание* – отдельным задачам предоставляется преимущественное право перейти в состояние ВЫПОЛНЕНИЯ.
- *Фиксированные приоритеты* – являются величиной постоянной на всем жизненном цикле процесса.
- *Динамические приоритеты* – изменяются в зависимости от некоторых условий в соответствии с определенными правилами. Для реализации динамических приоритетов необходимы дополнительные затраты, но их использование предполагает более справедливое распределение процессорного времени между процессами.

Линейные бесприоритетные дисциплины

- FCFS (First-Come, First-Served)
  - обслуживание процессов в порядке поступления
- SJF (Shortest-Job-First)
  - обслуживание самого короткого процесса первым без прерывания при поступлении более короткого процесса
- SRTF (Shortest-Remaining-Time-First)
  - обслуживание самого короткого процесса первым с прерыванием при поступлении процесса более короткого, чем остаток выполняющегося

First-Come, First-Served (очередь)

- FCFS (или FIFO) – самая простая дисциплина обслуживания, в соответствии с которой процессы получают доступ к процессору в порядке поступления.
- FCFS – реализует невытесняющую многозадачность.

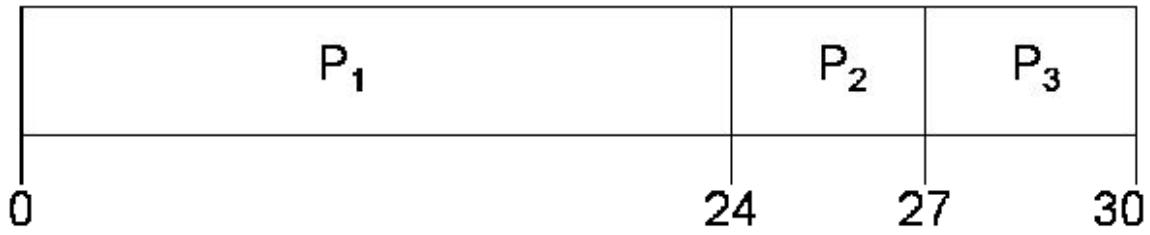


Пример FCFS – Вариант 1

Процесс	Время активности
P_1	24
P_2	3
P_3	3

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

Распределение процессора между процессами



Пример FCFS – Вариант 2

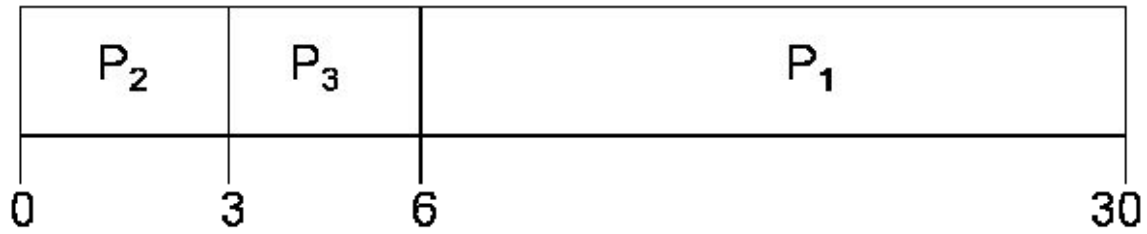
Процесс	Время активности
P_2	3
P_3	3
P_1	24

- Время ожидания процессов:

  $P_1 = 6; P_2 = 0; P_3 = 3.$
- Среднее время ожидания:

  $(6 + 0 + 3)/3 = 3.$

Распределение процессора между процессами



Shortest Job First

- Эффект, продемонстрированный примерами FSFS, носит название эффекта сопровождения (convoy effect) – увеличение среднего времени ожидания процессов в случаях, если короткий процесс обслуживается после долгого процесса.
- Для решения этой проблемы появилась стратегия Shortest Job First (SJF, обслуживание самого короткого задания первым) – стратегия диспетчеризации процессора, при которой процессор предоставляется в первую очередь наиболее короткому процессу из имеющихся в системе.

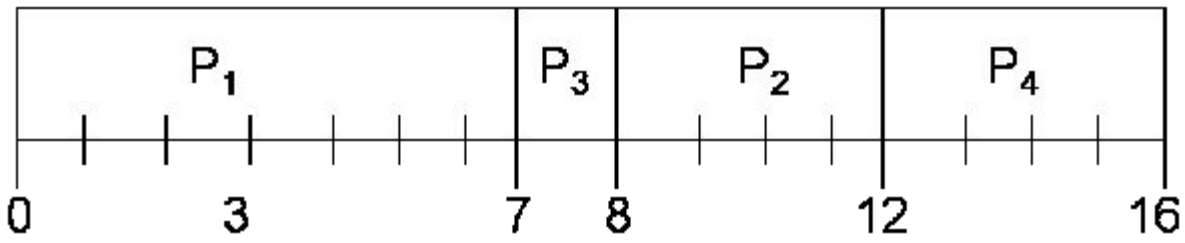
Пример SJF

Процесс	Время появления	Время активности
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- Среднее время ожидания:

  (0 + 6 + 3 + 7)/4 = 4.

Распределение процессора между процессами



Shortest-Remaining-Time-First

- Обслуживание самого короткого процесса первым.
- Если приходит новый процесс, время активности которого меньше, чем оставшееся время активного процесса, – прерывание активного процесса.
- Достоинство – сокращение среднего времени ожидания.
- Недостаток – прерывание активного процесса будет приводить к дополнительным издержкам, связанным со сменой контекста процессора.

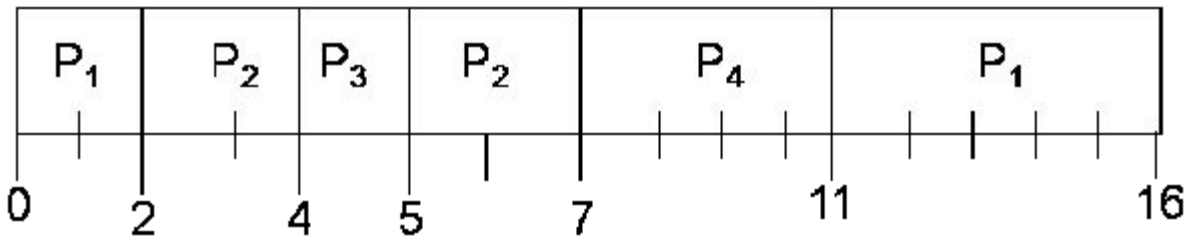
Пример SRTF

Процесс	Время появления	Время активности
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- Среднее время ожидания:

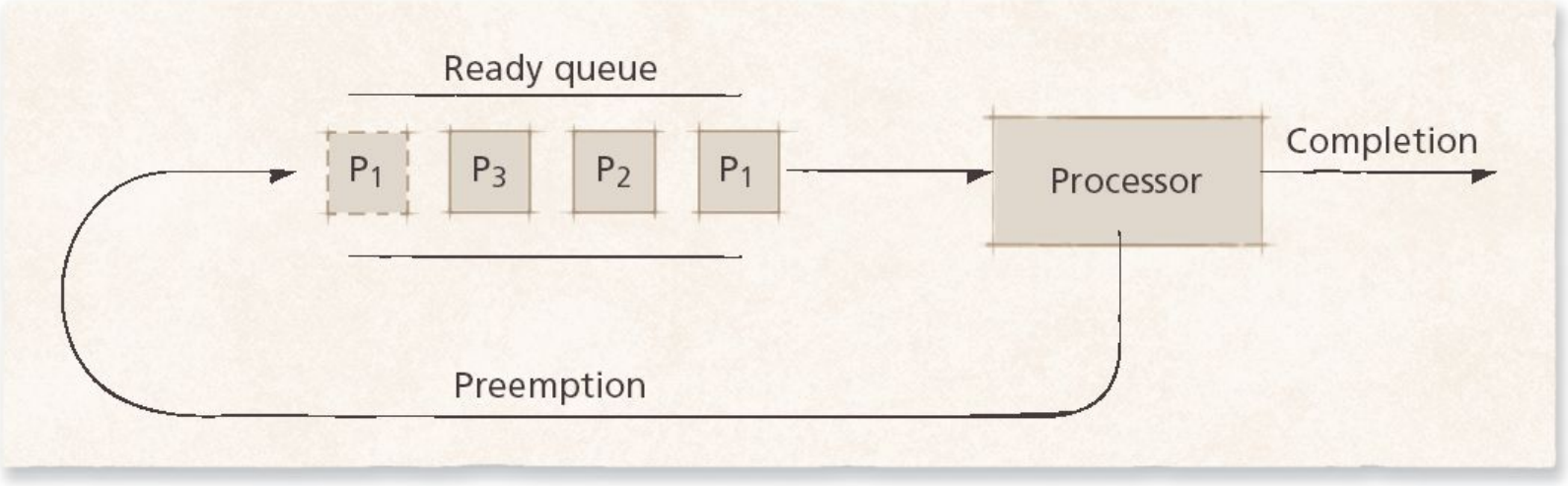
  (9 + 1 + 0 + 2)/4 = 3.

Распределение процессора между процессами



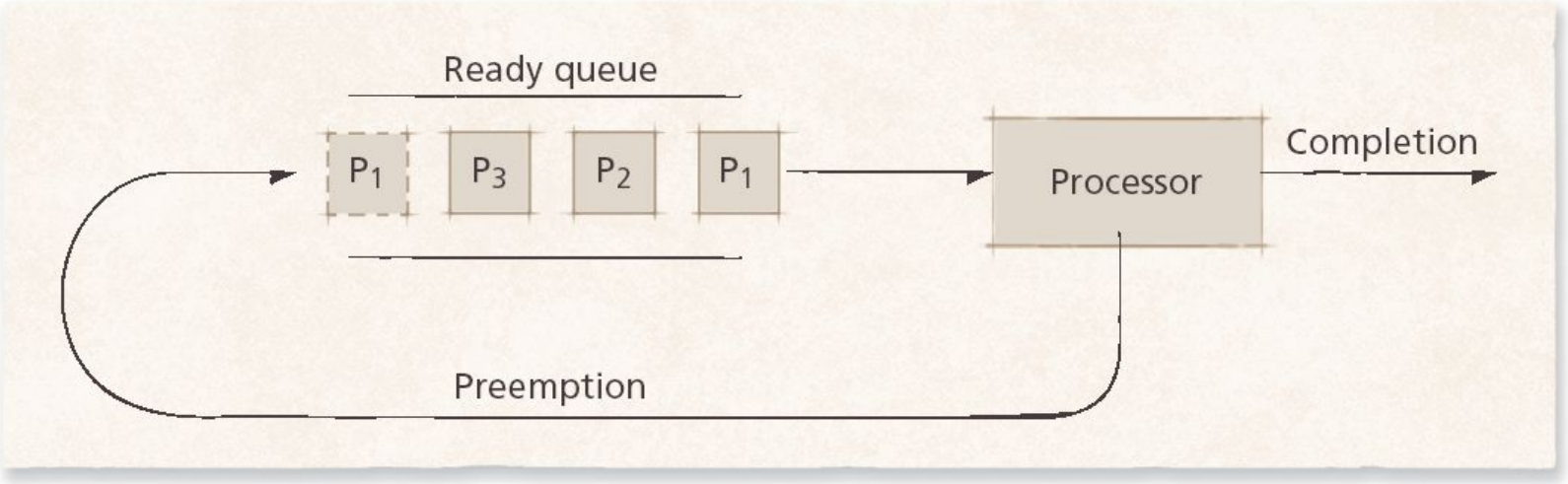
Квантование времени (1)

- Квантование времени (или Round Robin) – бесприоритетная циклическая дисциплина обслуживания:
  - каждый процесс (поток) получает небольшой квант процессорного времени, обычно – 10-100 миллисекунд;
  - после того, как это время закончено, процесс прерывается и



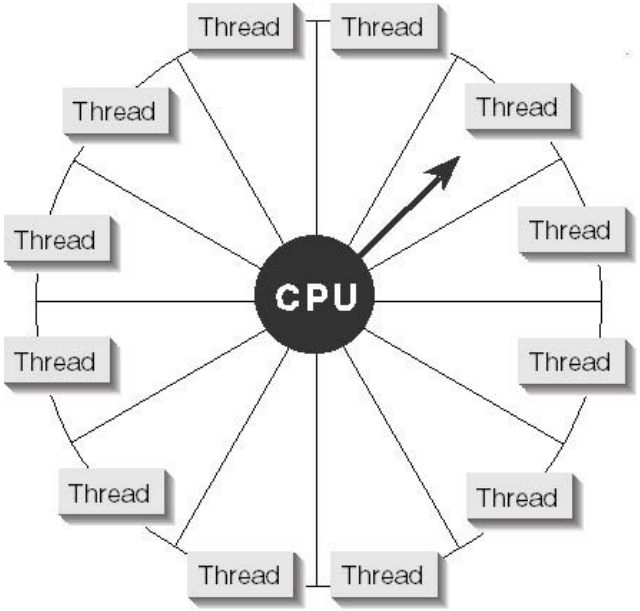
Квантование времени (2)

- Квантование времени реализует вытесняющую многозадачность – 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 квантования времени

- Производительность квантования времени зависит от q:
  - если q велико, то стратегия фактически эквивалентна стратегии FCFS;
  - если q мало, то q должно быть большим, чем время переключения контекста процессора, иначе слишком велики окажутся накладные расходы на переключения с одного процесса на другой.



- Обычно стратегия квантования времени имеет худшую пропускную способность, чем SJF, но лучшую реактивность.

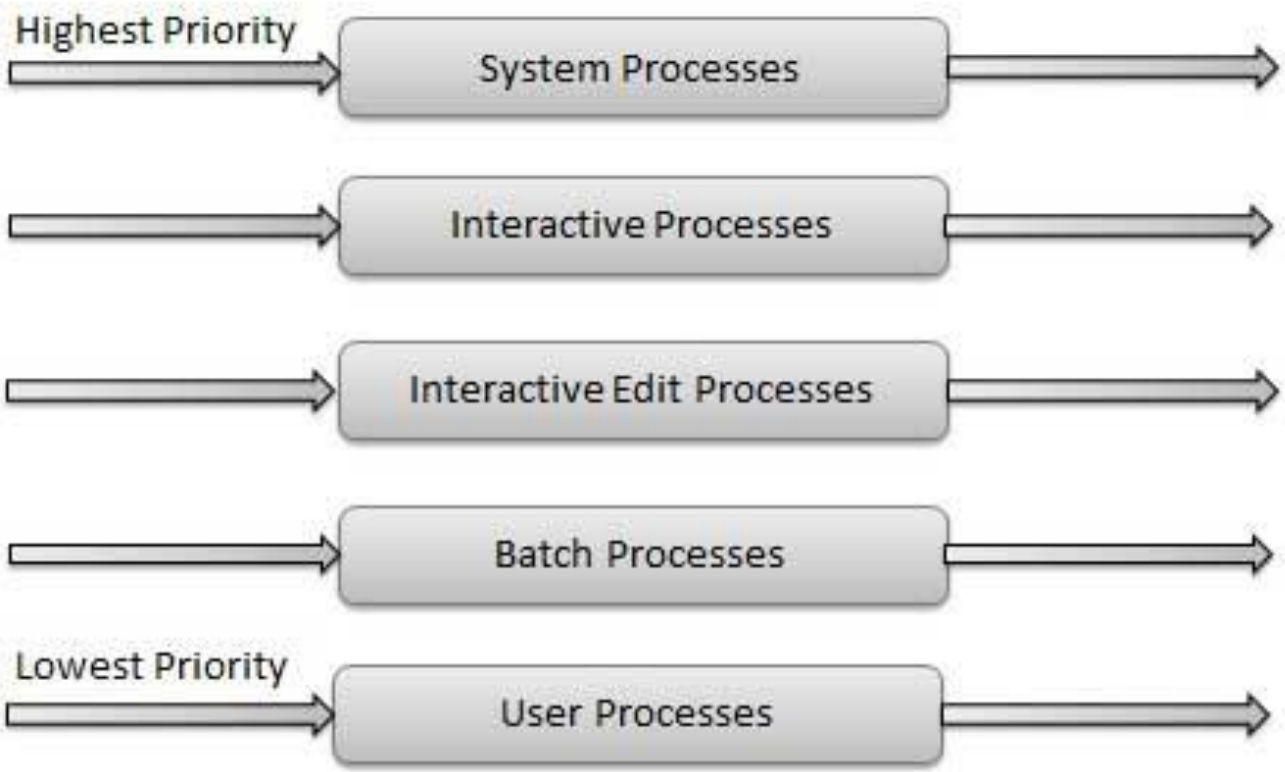
Многоуровневая очередь

- Поскольку процессы в системе могут иметь различную специфику (например, пакетные и интерактивные), то на практике в ОС очередь готовых процессов может быть разделена на две очереди (по сути это уже 2 уровня приоритетов):
  - основная (интерактивные процессы);
  - фоновая (пакетные процессы).
- Каждая очередь имеет свой собственный алгоритм диспетчеризации:
  - основная – квантование времени;
  - фоновая – FCFS.

Виды диспетчеризации между очередями

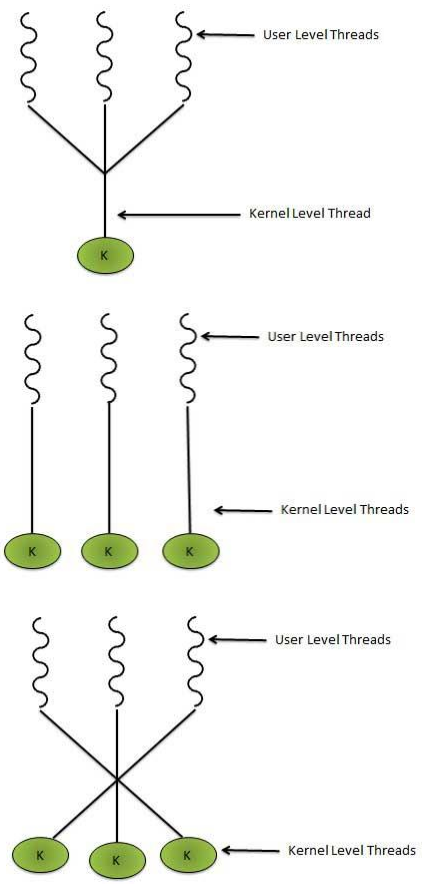
- При данной смешанной стратегии необходима также диспетчеризация между очередями, т.е. стратегия выбора процессов из той или иной очереди.
- Различаются следующие виды диспетчеризации между очередями:
  - **с фиксированным приоритетом** – обслуживание всех процессов из основной очереди, затем – из фоновой. При этом имеется вероятность «голодания» фоновых процессов.
  - **выделение отрезка времени** – каждая очередь получает некоторый отрезок времени ЦП, который она может распределять между процессами; например, 80% на основную очередь; 20% на фоновую очередь.

Пример многоуровневой очереди



Модели многопоточности

- **Многие к одному (Many-to-One)**
  - Несколько потоков пользовательского уровня отображаются в один поток ядра (многопоточность на уровне пользователя)
- **Один к одному (One-to-One)**
  - Каждый поток пользовательского уровня отображается в один поток ядра (однопоточность или многопоточность на уровне ядра) – Windows 2000+
- **Многие ко многим (Many-to-Many)**
  - Несколько потоков пользовательского уровня могут отображаться в несколько потоков ядра (комбинация 2-х предыдущих вариантов) – Solaris, Windows 2000+ (fibers)



Многопоточность на уровне пользователя

- можно реализовать в ОС, не поддерживающей потоки без каких-либо изменений в ОС;
- высокая производительность, поскольку процессу не нужно переключаться в режим ядра и обратно;
- управление потоками возлагается на программу пользователя, которая может использовать собственные алгоритмы планирования потоков с учетом их специфики.
- блокировка одного потока процесса приводит к блокировке всех потоков в рамках одного процесса;
- приложение не может работать в многопроцессорном режиме;
- невытесняющая многопоточность в рамках процесса – процессор не может быть принудительно отдан другому потоку в рамках одного процесса;
- внутри одного потока нет прерываний по таймеру, поэтому невозможно управлять потоками в режиме квантования времени.

Многопоточность на уровне ядра

- возможно планирование работы нескольких потоков одного и того же процесса на нескольких процессорах;
- реализуется мультипрограммирование в рамках всех процессов (в том числе одного);
- при блокировании одного из потоков процесса ядро может выбрать другой поток этого же (или другого процесса);
- процедуры ядра могут быть многопоточными.
- необходимость двукратного переключения режима пользователь – ядро, ядро – пользователь для передачи управления от одного потока к другому в рамках одного и того же процесса.