В системах разделения времени процессорное время квантуется
Квант – это отрезок времени, определенный в системе, и зависит от базовой частоты тактового генератора, т к эта частота определяет интенсивность выполнения действий в системе, выполнения команд, так как техника цифровая и работает от управления импульсами.
Компьютер выполняет последовательность команд, и каждая команда выполняется за определенное количество тактов, причем разные команды могут выполнятся за разное количество тактов.
В больших системах квант мог иметь разную длительность. То есть разные процессы могут получать кванты разной длительности. В персональных компьютерах этого нет.
Функции обработчика прерывания от системного таймера в системах разделения времени
Unix: По тику: декремент кванта; инкремент счётчика реального времени; инкремент счетчика процессорного времени, полученного текущим процессором; инкремент счетчика времени с момента запуска системы; декремент счетчиков времени отложенных вызовов (при достижении каким-либо счетчиком нуля установка флага для запуска обработчика отложенного вызова).
По главному тику: инициализация отложенного вызова функций планировщика; вызов pagedaemon и swapper; декремент счетчиков времени до посылки сигналов: SIGALRM, SIGPROF, SIGVTALRM.
По кванту: посылка сигнала SIGXCPU текущему процессу, если он превысил выделенный для него квант процессорного времени.
Windows: По тику: декремент кванта; инкремент счётчика реального времени; декремент счетчиков времени отложенных вызовов (при достижении каким-либо счетчиком нуля установка флага для запуска обработчика отложенного вызова).
По главному тику: инициализация диспетчера настройки баланса путем освобождения объекта «событие», на котором он ожидает;
По кванту: инициализация диспетчеризации потоков путем постановки соответствующего объекта в очередь DPC.
Основной задачей обработчика прерывания от системного таймера является **декремент кванта**, т е система должна получить информацию о том, что квант, выделенный конкретному процессу, истек.
При этом основной задачей системы при истечении кванта является передача кванта другому процессу. Но в персоналках имеется следующая особенность: если пользователь работает с приложением, например, у него открыто окно, то получится так что именно этот процесс, с которым в текущий момент работает пользователь, будет иметь всегда наивысший приоритет. Поэтому квант будет выдан этому же процессу.
Под системным таймером понимается **аппаратная подсистема вычислительной системы на основе кварцевого генератора** и специальных аппаратных схем с помощью которого выделяются так называемые тики.
В наших системах кванты измеряются тиками, и они в системах называются джифис. Джифис – это просто тик. Кроме тиков, есть главные тики, их в системе может быть несколько.
Таймер — это единственное независимое от процессора устройство, на которую можно повестись инициализацию периодически выполняемых действий в системе.
Прерывание от системного таймера — это аппаратное прерывание и самое

высокоприоритетное среди аппаратных прерываний. Т е обработчик прерывания от системного таймера вытеснит любую работу с выполнения. Выше только power – прерывания, которое возникает при падении напряжения. Когда падает напряжение основная задача системы сохранить работоспособность, для этого установлены минимально необходимые действия для сохранения соответствующей информации.
И когда выполняется данный обработчик, на процессоре на котором выполняется он, все прерывания запрещены, т е ничего другое выполняться не может. По данной линии IRQ прерывания запрещены на всех процессорах. Поэтому такой обработчик прерывания должен завершаться как можно быстрее, иначе это будет влиять на отзывчивость системы. **Отзывчивость системы** — это то насколько быстро системы реагирует на различные события.
Что может сделать обработчик прерывания от системного таймера? Он может инициализировать как отложенное действие работу планировщика, т е **все действия, которые инициализируются обработчиком прерывания от системного таймера являются отложенными**. Таймер не может выполнять в системе все функции, он так никогда не завершится.
В Windows есть dpc different procedure call – отложенный вызов процедуры. у них есть очередь dpc.
в Юниц в очередь ставятся дескрипторы процессов, потому что процесс — это абстракция. А что не абстракция? – структура struct task_struct – дексриптор
В ядре Windows нет приоритетов. ОС в стадии выполнения это супервизор.
Супервизор Windows выполняется на нулевом уровне приоритета, а внутри ядра все действия выполняются по IRQL. И у них есть tpc dispatch IRQL – диспетчеризация. А выше tpc dispatch все аппаратные прерывания – девайсы.
Диспетчеризация - выделение процессорного времени

В Unix для ядра выделен диапазон приоритетов. Обработчик прерывания от систем будет выполнятся на очень высоком уровне приоритета в диапазоне приоритетов ядра, потому что это код ядра.
Например, по главному тику вызывается pagedaemon. Его задача – заранее освобождать фреймы физической памяти, то есть вытеснять страницы по алгоритму LRU
Что еще должно заранее делаться – пересчитывать приоритеты, то есть приоритеты также пересчитываются по главному тику, заранее, чтобы, когда квант истек уже было понятно какой процесс имеет наивысший приоритет. Истек квант, и основная задача системы выделить квант процессу который находится первым в очереди готовых процессов по диаграмме состояний
Инициализировать отложенное действие можно с помощью сигнала, это надо вынуть из текста, в помощь изменения статуса

Пересчет динамических приоритетов

Единицей декомпозиции в системе является процесс, потому что именно процесс является владельцем ресурсов. Ресурсы запрашивают потоки. И процессу при его создании назначается приоритет и этот приоритет является базовым приоритетом для приоритетов потока. **То есть потоки имеют относительные приоритеты,** относительно базового приоритета процесса.

Пересчитываться могут только пользовательские приоритеты.

Для чего пересчитывать приоритеты? Чтобы исключить бесконечное откладывание, поэтому при пересчете приоритетов учитывается процессорное время полученное процессом и время простоя процесса в очереди готовых процессов и это четко видно в вахалии там есть формула

Сейчас уже в линукс можно менять планировщик, потому что предусмотрено их несколько. Для того чтобы пользователь мог выбрать планировщик, наиболее подходящий к выполнению его задач.

В Windows сейчас у основной алгоритм это “самый справедливый планировщик” – этот планировщик построен так чтобы все процессы получали примерно равное процессорное время, в зависимости от типа процесса, потому что процессы в наших системах разные

Windows также пересчитывают приоритеты, так же учитывается время простоя, но делается это путем сканирования очереди готовых процессов и в этой очереди ищется процесс который находился в этой очереди какой то заданный промежуток времени, такому процессу приоритет сразу резко повышается и он перемещается в начало списка. При этом в тексте СР упор делается на потоки.

Unix писался как системы разделения времени (это само собой разумеется что это много процессная система)

Алгоритм многоуровневые очереди для систем разделения времени. Какие процессы поступают в самую приоритетную очередь по этому алгоритму? – Вновь созданные и/или завершившие ожидание завершения операции ввода/вывода, они были блокированы в ожидании завершения операции ввода-вывода. То есть при пересчете приоритета учитывается не только процессорного времени получил процесс или сколько простаивал в очереди готовых процессов, но также учитывается на чем процесс был блокирован. Процесс блокируется не только на устройствах ввода-вывода он блокируется на объектах ядра, например, семафорах, что тоже повышает его приоритет

В Unix повышение приоритета процесса связано я приоритетом сна, то есть с приоритетом того, на чем процесс был блокирован.