|  |
| --- |
| 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  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. НАЧИНАТЬ С ЭТОГО НЕЛЬЗЯ.  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. ЭТО В НАЧАЛО.  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 обьектах ядра, например, семафорах, что тоже повышает его приоритет  В СР есть табличка, которая демонстрирует насколько будет повышен приоритет процесса который был блокирован на разных внешних устройствах, на звуковой карточке, на мыши на клаве, на дискетке. А про объекты ядра написано просто в тексте  В Unix повышение приоритета процесса связано я приоритетом сна, то есть с приоритетом того на чем процесс был блокирован. Эта табличка есть в др книгах  Наши системы являются системами общего назначения, то есть они признаны решать широкий круг самых широких задач. В наших системах есть процессы реального времени – это аудио и видео, соответственно слух человека очень чувствителен в звуковым задержкам  Процессы реального времени должны иметь статический приоритет, windows сделали их динамическими  Видео: наш глаз запоминает изображения на 1/24 секунды, на этом построено все связанное с кино и т д  Написать базовую информацию о том насколько будет повышен приоритет  Обработчик прерывания от системного таймера (больше нигде это словосочетание не использовать). Мы говорили **счетчик реального времени**, **счетчик тиков**. У нас часы которые показывают поясное время – реальное время. В обьектах ядра есть таймеры – alarm.  Введение писать не надо, задание тоже. Сразу – функции обработчика прерывания от системного таймера в системах разделения времени  Заканчиваем ВЫВОДАМИ, дель достигнута писать не надо. Эти выводы должны быть короткими, с 1/4 страницы не более, несколько фраз.  Титульник нужен, нумерация желательна |

В Вахалии о потоках ничего нет, но мы понимаем что современный Unix многопоточный, монолитное ядро многопоточное. Windows написан много проще, поэтому изначально это было монолитное многопоточное ядро но еще и оно обьектно-ориентированное. ООП очерь тяжелый – смысл его использования – сокращение времени разработки.

Выделенная квота, но почему они назвали это квотой – поскольку процесс может быть вытеснен с выполнения, то его квант уменьшается на уже полученное время, т е не так что его вытеснили и потом он получает опять тот же квант – нет. Из длительности кванта вычитается уже полученное время, поэтому они это называют квотой.

Квант это как набор джифисов, то есть коли

Квота это весь квант. Будем называть квантом, а дальше будем говорить об остатке кванта. Диаграммы последовательности вызовов и ответа

Процессору все равно он просто декрементирует квант

Массивы структур обьявляем в скелетоне, в стабе создаем набор семафоров

pagedaemon – это в unix bsd

linux – kthread daemon

2.4. Выполнение в режиме ядра

Существуют три различных типа событий, которые могут перевести систему в режим ядра. Это —

* **прерывания устройств** (ицбеггирй$),
* исключительные ситуации или просто исключения (ехсерНоп5), а также
* ловушки (гар) или программные прерывания (зоЁ\аге пщеггир(5).

Каждый раз, когда ядру возвращается управление, оно обращается к **таблице диспетчеризации (dispatch table),** содержащей адреса низкоуровневых процедур обработки событий. Перед вызовом соответствующей процедуры **ядро частично сохраняет состояние прерванного процесса** (например, указатель команд и слово состояния процессора) в стеке ядра для этого процесса. После завершения работы процедуры ядро восстанавливает состояние процесса и изменяет режим его выполнения на прежнее значение. Прерывание может произойти и в тот момент, когда система уже находится в режиме ядра, в таком случае она останется в нем и после обработки прерывания.

Прерывания — это асинхронные события, происходящие в периферийных устройствах, таких как диски, терминалы или аппаратный таймер. Так как прерывания не зависят от текущего процесса, они должны обрабатываться в системном контексте, при этом доступ в адресное пространство или область и процесса им не требуется. По этой же причине прерывания не должны производить блокировку, так как они могут заблокировать произвольный процесс.

2.4.2. Обработка прерываний

Функция, запускаемая для обслуживания прерывания, называется обработчиком прерывания, или процедурой обслуживания прерывания. Обработчик работает в режиме ядра и системном контексте. Так как прерванный процесс обычно не имеет никакого отношения к произошедшему в системе прерыванию, обработчик не должен обращаться к контексту процесса. По той же причине он также не обладает правом блокировки. Однако прерывание оказывает некоторое влияние на выполнение текущего процесса. Время, потраченное на обработку прерывания, является частью кванта времени, отведенного процессу, даже если производимые действия не имеют ни малейшего к нему отношения. Так, обработчик прерываний системного таймера использует тики (промежутки времени между двумя прерываниями таймера) текущего процесса и потому нуждается в доступе к его структуре ргос. Важно отметить, что контекст процесса защищен от доступа обработчиками прерываний не полностью. Неверно написанный обработчик может причинить вред любой части адресного пространства процесса.

В системах UNIX каждому типу прерывания принято **назначать уровень приоритета прерывания** (ицеггирЕ рпогку 1еуе1, 1р|). В первых реализациях системы уровень 1р| находился в пределах от 0 до 7. В ОС ВЗО значение 11 возросло до 0-31. Регистр состояния процессора обычно содержит битовые поля, в которых хранится текущий (а иногда и предыдущий) тр. Номера приоритетов прерываний не одинаковы, так как зависят не только от конкретной реализации системы (МХ, но и от различия в архитектуре оборудования. В некоторых системах 1] 0 означает низший уровень приоритета, тогда как в иных это значение может оказаться наивысшим. Для облегчения создания процедур обработки прерываний и драйверов устройств в системах МХ представлен набор триггеров для блокировки и разблокирования прерываний. Однако в различных реализациях системы для одних и тех же целей используются различные триггеры. В табл. 2.1 показаны некоторые триггеры, применяемые в 4.3В$О и ЗУБА.

стр 81

Ядро использует триггеры, приведенные в табл. 2.1 для повышения уровня ipl и блокировки всех прерываний.

табл. 2.1 Установка уровней приоритетов прерываний в системах

Глава 5 Планирование процессов

Система UNIX изначально создавалась как ОС разделения времени, что означает возможность одновременного выполнения в ней нескольких процессов.

В системах UNIX эмулируется одновременность работы при помощи чередования процессов на основе принципа разделения времени. Планировщик предоставляет процессор каждому процессу системы на небольшой период времени, после чего производит переключение на следующий процесс. Такой период называется квантом времени