Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ДИСЦИПЛИНА «Операционные системы»

Лабораторная работа № 1(2)

Тема «Изучение функций системного таймера в защищенном режиме. Пересчет динамических приоритетов для операционных систем семейств Windows и UNIX/Linux»

Студент Шиленков А. А.

Группа ИУ7-55Б

Оценка (баллы) _____

Преподаватель Рязанова Н.Ю.

Москва.
2020 г.

Задание

В первой части необходимо изучить функции прерываний от системного таймера по литературе. Самостоятельно оформить и предоставить отчет, в котором перечисляются каждая из функций системного таймера для операционных систем семейств Windows и UNIX/Linux. Распределение строится таким образом: по тику, по главному тику и по кванту.

Во второй части необходимо ищучить особенности пересчета динамических приоритетов для операционных систем, указанных ранее. Так же необходимо оформить отчет. Информация должна опираться на актуальную информацию.

Функции обработчика прерываний от системного таймера.

Необходимо ввести некоторые определения.

Тик — период времени между двумя последовательными прерываниями таймера, компьютерная единица времени.

Главный тик — период времени, равный некоторому числу тиков. Число таких типов зависит от конкретной системы.

Квант — количество времени, которое предоставляется процессу планировщиком для выполнения в процессоре.

Нас интересуют два конкретных семейства:

1. **Windows**

По тику:

(1) Инкрементирует счетчик системного времени

(2) Декрементирует остаток кванта текущего потоками

(3) Декрементирует счетчик отложенных задач

(4) Добавление в очередь DPC обработчика ловушки профилирования ядра

По главному тику:

(1) Инициализация диспетчера настройки баланса (освобождение «событие» каждую секунду)

По кванту:

(1) Инициализация диспетчеризации потоков (добавление соответствующего DPC в очередь)

2. Unix/Linux

По тику:

(1) Инкремент часов и других таймеров системы

(2) Инкрементирует счетчик использования процессора текущем процессом

(3) Инкрементирует счетчик тиков аппаратного таймера

(4) Декременирует счетчик времени до отправления на выполнение отложенного вызова

(5)Декременирует квант текущего потока

По главному тику:

(1) Добавление в очередь отложенных вызовов функций планировщика

(2) Пробуждение системных процессов swapper и pagedeamon

(3) Декремент счетчиков времени, оставшегося до отправления одного из сигналов: SIGALARM, SIGPROF, SIGVTALARM

SIGALARM — сигнал, посылаемый процессу по истечении заданного промежутка времени

SIGPROF — сигнал, посылаемый процессу по истечении времени заданном в таймере профилирования

SIGVTALARM — сигнал, посылаемый процессу по истечении времени, заданного в «виртуальном» таймере.

По кванту:

(1) При истечении выделенного текущему процессу кванта — отправка сигнала SIGXCPU этому процессу.

Пересчет динамический приоритетов

Динамически пересчитываться могут только приоритеты пользовательских процессов.

Пересчет динамических приоритетов в Windows

В Windows при создании процесса, ему назначается базовый приоритет. Относительного базового приоритета процесса потоку назначается относительный приоритет.

В данной системе реализуется приоритетная, вытесняющая система планирования.

Раз в секунду диспетчер настройки баланса сканирует очередь готовых потоков. Если там обнаружены потоки, которые ожидают времня выполнения дольше 4 секунд, то диспетчер настройки баланса повышает их приоритет до 15. Квант истек — приоритет снижается до базового. Если поток не был завершен за квант времени или был вытеснен потоком с более высоким приоритетом, то после снижения поток попадает в очередь готовых потоков. В следующий проход сканирование возобновляется с того места, где было прервано.

Используются 32 уровня приоритета — от 0 до 31.

- от 16 до 31 — реальное время

- от 1 до 15 — изменяющиеся уровни
- 0 — резерв для потока обнуления страниц

Уровни приоритета потоков назначаются от Windows API и ядра Windows.

Сначала Windows API систематизирует процессы по классу приоритета, который им присваивается:

- Реального времени (4)
- Высокий (3)
- Выше обычного (7)
- Обычный (2)
- Ниже обычного (5)
- Простой (1)

Далее назначается относительный приоритет отдельных потоков внутри процессов. Номера применяются к базовому приоритету процесса:

- Критичный по времени (15)
- Наивысший (2)
- Выше обычного (1)
- Обычный (0)
- Ниже обычного (-1)
- Наименьший (-2)
- Простой (-15)

Базовый приоритет потока наследуется от базового приоритета процесса.

Решение по планированию принимается на основе текущего приоритета. Система вправе при определенных обстоятельствах на короткие периоды времени повышает приоритет потоков в динамическом диапазоне. В диапазоне реального времени Windows этого не делает, оставляет неизменными текущий и базовый приоритеты.

Текущий приоритет потока в динамическом диапазоне — от 1 до 15, может быть повышен планировщиком когда:

- вследствие события планировщика или диспетчера
- повышение приоритета владельца блокировки
- повышение приоритета после завершения ввода\вывода

  Повышение приоритета 1 — жесткий лиск, привод CD, параллельный порт, видео устройство

  Повышения приоритета 2 — сеть, почтовый слот, именованный канал, последовательный порт

  Повышение приоритета 3 — клавиатура, мышь

  Повышение приоритета 4 — звуковое устройство

  Сокращаются задержки
- повышение приоритета вследствие ввода из пользовательского интерфейса
- повышение приоритета вследствие длительного ожидания ресурса исполняющей системы

- повышение приоритета, когда готовый к выполнению поток не был запущен в течение длительного времени
- повышение приоритета, когда центральный процессор перегружен

Распределение ресурсов процессора зависит от приоритета потока.

Типичные ситуации планирования:

- Переключение самостоятельно. Поток может отказаться от использования процессора путем входа в состояние ожидания какого-то объекта.
- Вытеснение, когда поток с более низким приоритетом вытесняется , а поток с более высоким приоритетом готов к выполнению.
- Истечение кванта времени. У выполняемого потока кончился квант, определяется целесообразность снижения приоритета потока, а затем определяется, нужно ли спланировать выполнение на процессоре уже другого потока.
- Завершение выполнения потока. Переход из состояния выполнения в состояние завершения.
- Ожидание, простой. При том условии, когда у центрального процессора нет потоков, готовых к выполнению. Запускается поток простоя.

Пересчет динамических приоритетов в UNIX/Linux

Классическое ядро UNIX является строго невытесняемым. Если процесс выполняется в режиме ядра, то ядро не заставит этот процесс уступить процессорное время какому-либо более приоритетному процессу.

Современные ядра Linux с версии 2.5 являются полностью вытесняемыми, для обеспечения работы процессов реального времени.

В данных системах планировщик предоставляет каждому процессу системы квант времени, по истечению которого происходят переключения на следующий процесс — это так же и система разделения времени.

Процесс в режиме ядра может быть вытеснен более приоритетным процессом в режиме ядра. Это нужно для того, чтобы модно было обслуживать процессы реального времени такие, как аудио или видео.

В современных системах Unix/Linux ядро является вытесняемым, планировщик всегда выбирает процесс с наивысшим приоритетом, приоритет процесса динамически изменяется в зависимости от использования вычислительных ресурсов, времени ожидания запуска и текущего состояния процесса.

Если процесс готов к запуску и имеет наивысший приоритет, планировщик приостановит выполнение текущего процесса, если он имеет более низкий приоритет, даже если его квант не израсходован.

Структура proc для описания приоритета:

p_pri — Текущий приоритет планирования

p_usrpri — Приоритет режима задачи

p_cpu — Результат последнего измерения использования процессор

p_nice — Фактор «любезности», устанавливаемый пользователем.

В других версиях эта структура может иметь другое название и\или другой вид.

Планировщик использует p_pri для того, чтобы решить, какой процесс направить на выполнение. У процесса, находящегося в режиме задачи — p_pri = p_usrpri, т.е. идентичны. Значение текущего приоритета может быть повышено планировщиком для выполнения процесса в режиме ядра, в случае чего p_usrpri будет использоваться для хранения приоритета, который будет назначен процессу при возврате в режим задачи.

При создании процесса p_cpu инициализируется нулем. На каждом тике он инкрементируется.

Фактор «любезности» - это целое число от 0 до 39 (20 по умолчанию), что приводит к уменьшению приоритета процесса. Он может быть изменен суперпользователем с помощью системного вызова nice.

Приоритет задается любым целым числом в диапазоне от 0 до 127. Чем меньше число, тем более приоритетный процесс. Приоритеты от 0 до 49 зарезервированы для ядра. Прикладные процессы — с 50 до 127. В первую очередь выполняются процессы с большим приоритетом. Если возникает ситуация, когда у процессов одинаковые приоритеты, тогда они выполняются в течении кванта времени циклически друг за другом.

Ядро системы связывает приоритет сна с событием или ожидаемым ресурсом, из-за которого процесс может быть блокирован. Когда процесс проснулся после блокировки, ядро устанавливает значение значение текущего приоритета процесса равным приоритету сна. Это позволяет в моменты необходимости предоставить вычислительные ресурсы. К примеру — быстрое завершение системного вызова, выполнение которого может блокировать некоторые системные ресурсы.

Системные приоритеты сна

Событие	Приоритет 4.3BSD UNIX	SCO UNIX
Ожидание загрузки в память сегмента\страницы	0	95
Ожидание индексного дескриптора	10	88
Ожидание ввода\вывода	20	81
Ожидание буфера	30	8
Ожидание терминального ввода	30	75
Ожидание терминального вывода	30	74
Ожидание завершения выполнения	30	73
Ожидание события	40	65

Задача планировщика — распределить вычислительный ресурс между конкурирующими процессами.

Приоритет в режиме задачи зависит от двух факторов

- «Любезности». Чем выше любезность, тем ниже приоритет.
- Последней измеренной величины использования процессора. Каждый тик инкрементируется.

Каждую секунду ядро системы пересчитывает текущие приоритеты процессов, готовых к запуску (процессов, находящихся в режиме задачи). Уменьшается значение p_cpu каждого процесса исходя из формулы:

$$decay = \frac{2 \cdot load\_average}{2 \cdot load\_average + 1}$$

- фактор полураспада, где load_average — это среднее количество процессов, находящихся в состоянии готовности к выполнению, за последнюю секунду.

Сам пересчет приоритетов происходит по формуле:

$$p\_usrpri = PUSER + \frac{p\_cpu}{2} + 2 \cdot p\_nice$$

, где PUSER — базовый приоритет в режиме задачи.

Если процесс в последний раз, до вытеснения другим процессом, использовал много процессорного времени, то его результат процессорного времени — p_cpu, будет увеличне, что приведет к понижению приоритета.

Чем дольше процесс простаивает в очереди, тем выше его приоритет, что предотвращает бесконечное откладывание, свойственное прошлым системам. Алгоритм обеспечивает более вероятный выбор планировщиком интерактивных процессов по отношению к вычислительным, поскольку процесс, расходующий ьольшую часть времени выполнения на ожидания ввода-вывода остается с высоким приоритетом. Это важно в работе пользователя.

Заключение

ОС Windows и Unix\Linux — являются системами разделения времени, в которых обработчик системного таймера выполняет схожие основные функции, а именно пересчет приоритетов, инкремент и декремент счетчиков, декремент кванта процессорного времени.

Классическое ядро UNIX является строго невытесняющим. Ядра Linux, начиная с версии 2.5, и ядра операционных систем Windows являются полностью вытесняющими для обеспечения работы процессов реального времени.

В Unix\Linux приоритет может пересчитываться динамически, в зависимости от «любезности» и времени работы, базового приоритета. Приоритеты ядра — фиксированные величины.

В случае с Windows при создании процесса, процессу назначается базовый приоритет, а потокам — относительный приоритет, который может быть пересчитан.

Литература

1. М. Руссинович, Д. Соломон Внутреннее устройство Windows

2. Ю. Вахалия. UNIX изнутри.