|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»\_\_ \_\_\_\_

**Лабораторная работа № 1.**По курсу «Операционные системы».  
Тема: «Функции обработчика прерываний от системного таймера и пересчёт динамических приоритетов».

Студент: Уласик Е.А.  
Группа: ИУ7-51Б.  
Преподаватель: Рязанова Н. Ю.

*2019 г.*

**Функции обработчика прерывания от системного таймера**

1. **В Windows**

**По тику**

* Инкремент счётчика системного времени
* Инкремент счётчика времени использования процессора текущим процессом;
* Декремент остатка кванта текущего потока

**По главному тику**

* Инициализация диспетчера настройки баланса путем постановки объекта DPC в очередь.

**По кванту**

* Инициализация диспетчера настройки баланса путем постановки объекта DPC в очередь.

1. **В UNIX/LINUX**

**По тику**

* Инкремент счетчика использования процессора текущим процессом;
* Инкремент таймеров системы;
* Декремент счетчика времени до отправления на выполнение отложенных вызовов, добавление отложенных вызовов на выполнение при достижении нулевого значения счётчика.

**По главному тику**

* Добавление в очередь на выполнение функций, относящихся к работе планировщика-диспетчера, таких как пересчёт приоритетов и действия, которые будут выполнены по истечении выделенного кванта;
* Декремент времени, оставшегося до отправления SIGALRM (истечение времени, предварительно заданного функцией alarm(), измеряет реальное время) или SIGPROF (истечении времени заданном в таймере профилирования, измеряет время выполнения самого процесса и время, проведённое в ожидании завершения системных вызовов) или SIGVTALRM (истечение времени, заданного в «виртуальном» таймере, измерение работы процесса в режиме задачи);
* Изменение состояние процессов swapper и pagedaemon со «sleep» на «runnable».

**По кванту**

* Если израсходован выделенный квант процессорного времени, текущему процессу посылается сигнал SIGXCPU.

В Windows длительность интервала таймера зависит от аппаратной платформы и определяется HAL, а не ядром. Например, этот интервал на большинстве многопроцессорных систем — около 15 мс и зависит от частоты тактового генератора процессора.

В UNIX продолжительность тика обычно составляет 10 мс. Во многих реализациях частота таймера хранится в специальной константе HZ, определенной в файле param.h.

**Пересчет динамических приоритетов (для пользовательских процессов)**

***Windows***

Ядро Windows не имеет центрального потока планирования. Вместо этого, когда поток не может больше выполняться, он сам вызывает планировщик, чтобы увидеть, не освободился ли в результате его действий поток с более высоким приоритетом планирования, который готов к выполнению. Если это так, то происходит переключение потоков. В Windows реализуется приоритетная, вытесняющая система планирования, при которой всегда выполняется хотя бы один работоспособный (готовый) поток с самым высоким приоритетом. Но поток может и не израсходовать свой квант времени, поскольку в Windows реализуется вытесняющий планировщик: когда готов к запуску другой поток с более высоким приоритетом, текущий выполняемый поток может быть вытеснен еще до окончания его кванта времени. У процесса имеется только одно значение приоритета – базовое, а у потока – два: базовое и текущее, планировщик принимает решение на основе текущего приоритета. Начальный базовый приоритет потока наследуется из базового приоритета процесса. Процесс по умолчанию наследует свой базовый приоритет у родительского процесса.

Планирование вызывается при следующих условиях:

1. Поток становится готовым к выполнению — например, поток был только что создан или только что освобожден от состояния ожидания
2. Поток выходит из состояния выполнения из-за окончания его кванта времени, его работа завершается, ему предоставляется возможность выполнения или он входит в состояние ожидания.
3. Изменяется приоритет потока либо по причине вызова системной службы, либо по причине того, что Windows сама изменяет значение приоритета.
4. Изменяется сходство процессора потока, и он больше уже не может быть запущен на том процессе, на котором выполнялся.

В Windows 32 уровня приоритета: от 0 до 31: шестнадцать уровней реального времени (от 16 до 31), шестнадцать изменяющихся уровней (от 0 до 15), из которых уровень 0 зарезервирован для потока обнуления страниц.

Уровни приоритета потоков назначаются исходя из двух разных позиций: одной от Windows API и другой от ядра Windows. Сначала Windows API систематизирует процессы по классу приоритета, который им присваивается при создании:

* Реального времени — Real-time (4),
* Высокий — High (3),
* Выше обычного — Above Normal (7),
* Обычный — Normal (2),
* Ниже обычного — Below Normal (5)
* Уровень простоя — Idle (1).

Затем назначается относительный приоритет отдельных потоков внутри этих процессов. Здесь номера представляют изменение приоритета, применяющееся к базовому приоритету процесса:

* Критичный по времени — Time-critical (15)
* Наивысший — Highest (2)
* Выше обычного — Above-normal (1)
* Обычный — Normal (0)
* Ниже обычного — Below-normal (–1)
* Самый низший — Lowest (–2)
* Уровень простоя — Idle (–15).

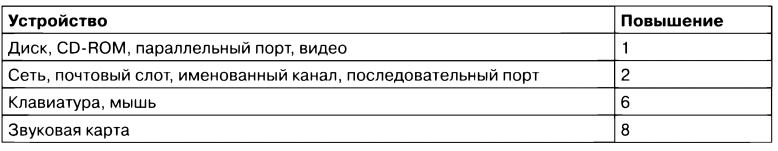
Потоки приложений обычно выполняются с приоритетами 1–15. Как правило, пользовательские приложения и службы запускаются с обычным базовым приоритетом (normal), поэтому их исходный поток чаще всего выполняется с уровнем приоритета 8.

Приоритет потока повышается, если:

* Повышение приоритета после завершения ввода/вывода

Windows даёт временное повышение приоритета при завершении определённых операций ввода-вывода, при этом потоки, ожидавшие ввода-вывода, имеют больше шансов сразу же запуститься и обработать то, что они ожидали. Рекомендуемые значения повышения приоритета указаны в таблице 1 (хотя подходящее значение для увеличения зависит от драйвера устройства):

Таблица 1 - Рекомендуемые значения повышения приоритета



Из таблицы видно, что запросы ввода-вывода к устройствам, гарантирующим наилучшую отзывчивость, имеют наиболее высокие значения повышения приоритета.

* Повышение приоритета потоков первого плана после ожидания

Когда поток в процессе первого плана завершает операцию ожидания объекта ядра, ядро повышает его текущий (но не базовый) приоритет на текущее значение переменной PsPrioritySeparation (какой процесс отнести к процессам первого плана, решает система управления окнами).

Смысл этого повышения заключается в улучшении отзывчивости интерактивных приложений.

* Повышение приоритета после пробуждения GUI-потока

Потоки-владельцы окон получают при пробуждении дополнительное повышение приоритета на 2 из-за активности системы работы с окнами.

* Если поток, готовый к выполнению, задерживается из-за нехватки процессорного времени.

Один раз в секунду этот поток сканирует очередь готовых потоков в поиске тех из них, которые находятся в состоянии ожидания (т. е. не были запущены) около 4 секунд. Если такой поток будет найден, диспетчер настройки баланса повышает его приоритет до 15 единиц и устанавливает квантовую цель эквивалентной тактовой частоте процессора при подсчете 3 квантовых единиц. Как только квант истекает, приоритет потока тут же снижается до обычного базового приоритета. Если поток не был завершен и есть готовый к запуску поток с более высоким уровнем приоритета, поток с пониженным приоритетом возвращается в очередь готовых потоков, где он опять становится подходящим для еще одного повышения приоритета, если будет оставаться в очереди следующие 4 секунды.

Повышение приоритета вступает в действие немедленно и может вызвать изменения в планировании процессора. Однако если поток использует весь свой следующий квант, то он теряет один уровень приоритета и перемещается вниз на одну очередь в массиве приоритетов. Если же он использует второй полный квант, то он перемещается вниз еще на один уровень, и так до тех пор, пока не дойдет до своего базового уровня (где и останется до следующего повышения). Повышение приоритета потока в Windows применяется только для потоков с приоритетом динамического диапазона (0-15). Но каким бы ни было приращение, приоритет потока никогда не будет больше 15. Таким образом, если к потоку с приоритетом 14 применить динамическое повышение на 5 уровней, то его приоритет станет равным только 15 (если приоритет потока равен 15, то повысить его нельзя).

***UNIX/LINUX***

Приоритет процесса/потока задается любым целым числом, лежащим в диапазоне от 0 до 139, то есть существует 140 уровней приоритета(для обычных потоков и потоков реального времени). Чем меньше такое число, тем выше приоритет. В Unix приоритеты от 0 до 49 зарезервированы для ядра, следовательно, прикладные процессы могут обладать приоритетом в диапазоне 50–139.

В фазе “пользователь” приоритет процесса имеет 2 составляющие: пользовательскую и системную. Значения этих составляющих задают поля дескриптора процесса p\_nice и p\_cpu. Начальное значение пользовательской составляющей равно константе NZERO (=20). Пользовательская составляющая может быть изменена системным вызовом nice с аргументом, определяющим величину изменения поля p\_nice в пределах от 0 до NZERO для непривилегированного процесса и от –NZERO до +NZERO для привилегированного. Начальное значение системной составляющей в фазе пользователь равно 0. Ее изменение зависит от времени использования процессора. Для формирования системной составляющей приоритета используются прерывания от аппаратного таймера. Каждое прерывание по таймеру увеличивает значение поля p\_cpu на 1. Результирующий приоритет процесса в фазе «пользаватель» определяется по формуле 1.

(1)

где P\_USER - константа, по умолчанию равная 50.

Каждую секунду ядро системы вызывает процедуру schedcpu(), запускаемую через отложенный вызов. Эта процедура уменьшает значения p\_cpu каждого процесса исходя из фактора полураспада, который определяется по Формуле (2).

(2)

где load\_average – среднее количество процессов, находящихся в состоянии готовности к выполнению, за последнюю секунду.

Планировщик всегда выбирает процесс с наивысшим приоритетом. Приоритет пользовательского процесса не является фиксированным и динамически изменяется системой 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 (с более низким приоритетом), даже если последний не "выработал" свой временной квант.

В UNIX структура proc содержит следующие поля, относящиеся к приоритетам:



Поле p\_pri используется планировщиком при выборе процесса для отправки его на выполнение. Планировщик использует значения p\_pri для хранения временного приоритета для выполнения в режиме ядра, а p\_usrpri для хранения приоритета, который будет назначен процессу при возврате в режим задачи. Процессу, ожидающему недоступного в данный момент ресурса, система определяет значение приоритета сна, связанного с событием, вызвавшим это состояние.

Поскольку приоритеты ядра выше, то такие процессы будут назначены на выполнение раньше, чем другие, функционирующие в режиме задачи. Благодаря этому системные вызовы могут завершать работу быстро, что является желательным, так как процессы во время выполнения вызова могут занимать некоторые ключевые ресурсы системы, не позволяя другим ими пользоваться. Когда процесс завершит выполнение системного вызова, его приоритет сбрасывается обратно в значение текущего приоритета в режиме задачи, произойдёт переключение контекста. Традиционное ядро UNIX не было вытесняющим, однако современные версии являются полностью преемптивным.

**Вывод**

И в ОС Windows, и в UNIX обработчик системного таймера выполняет схожие основные функции:

1. обновление системного времени.
2. уменьшение кванта процессорного времени, выделенного процессу.
3. запуск планировщика задач.
4. отправление отложенных вызовов на выполнение.

Это обусловлено тем, что обе операционные системы являются системами разделения времени с вытеснением и динамическими приоритетами.