|  |  |
| --- | --- |
| 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»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Отчет**

***к лабораторной работе №1 по курсу «Операционные системы»***

***по теме «Изучение функций прерывания от системного таймера и особенностей пересчета динамических приоритетов для ОС семейств Windows и Unix/Linux»***

|  |  |
| --- | --- |
| **Студент: Русинова Д. Э.**  **Группа: ИУ7-55б**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель: Рязанова Н. Ю.** |  |

Москва.

2020 г.

**Функции обработчика прерывания от системного таймера в защищённом режиме для операционных систем семейства Unix/Linux.**

**По тику.**

* Инкремент счетчика тиков аппаратного таймера.
* Декремент кванта текущего потока.
* Обновление статистики использования процессора текущим процессом: инкремент поля p\_cpu дескриптора текущего процесса на единицу, до максимального значения, равного 127.
* Обновление часов и других таймеров систем.
* Декремент счетчика времени до отправления на выполнение отложенных вызов. Если счетчик достиг нуля, то выставление флага для обработчика отложенных вызов.

**По главному тику.**

* Инициализация отложенных вызовов функций, относящихся к работе планировщика, таких как пересчет приоритетов.
* Пробуждение в нужные моменты системных процессов, таких как swapper и pagedaemon. Под пробуждением понимается инициализация отложенного вызова процедуры wakeup, которая перемещает дескрипторы процессов из списка «спящих» в очередь готовых к выполнению.
* Декремент времени, оставшегося времени до посылки одного из следующих сигналов:
  + SIGALARM – декремент будильника реального времени;
  + SIGPROF – измерение времени работы процесса;
  + SIGVTALARM – измерение времени работы процесса в режиме задачи.

**По кванту.**

* Посылка текущему процессу сигнала SIGXCPU, если тот превысил выделенную ему квоту использования процессора. По получению сигнала обработчик сигнала прерывает выполнение процесса.

**Функции обработчика прерывания от системного таймера в защищённом режиме для операционных систем семейства Windows.**

**По тику.**

* Инкремент счетчика системного времени.
* Декремент кванта текущего потока на величину, равную количеству тактов процессора, произошедших за тик. Если количество затраченных потоком тактов процессора достигает квантовой цели, запускается обработка истечения кванта.
* Декремент счетчиков времени отложенных задач.
* Если активен механизм профилирования ядра, то инициализация отложенного вызова обработчика ловушки профилирования ядра путем постановки объекта в очередь DPC: обработчик ловушки профилирования регистрирует адрес команды, выполнявшейся на момент прерывания.

**По главному тику.**

* Освобождение объекта «событие», которое ожидает диспетчер настройки баланса: по событию от таймера диспетчер настройки баланса сканирует очередь готовых процессов и повышает приоритет процессов, которые находились в состоянии ожидания дольше 4 секунд.

**По кванту.**

* Инициализация диспетчеризации потоков путем постановки соответствующего объекта в очередь DPC.

**И в ОС семейства Unix/Linux динамически пересчитываться могут только приоритеты пользовательских процессов.**

**Пересчет динамических приоритетов в операционных системах семейства Unix/Linux.**

Очередь готовых к выполнению процессов формируется согласно приоритетам процессов и принципу вытесняющего циклического планирования: процессы с одинаковыми приоритетами выполняются в течении кванта времени циклически друг за другом. Если процесс, имеющий более высокий приоритет, поступает в очередь готовых к выполнению, планировщик вытесняет текущий процесс и предоставляет ресурс более приоритетному.

В традиционных системах Unix ядро является строго невытесняющим, однако в современных системах Unix ядро является вытесняющим – процесс в режиме ядра может быть вытеснен более приоритетным процессом в режиме ядра. Ядро было сделано вытесняющим для того, чтобы система могла обслуживать процессы реального времени, такие как аудио и видео.

Приоритет представляет собой целое число из диапазона от 0 до 127. Чем меньше число, тем выше приоритет:

* в диапазоне от 0 до 49 находятся приоритеты ядра;
* в диапазоне от 50 до 127 – приоритеты прикладных задач.

Приоритеты ядра являются фиксированными величинами.

Приоритеты прикладных задач могут изменяться во времени в зависимости от следующих двух факторов:

* Фактор любезности – целое число в диапазоне от 0 до 39. Чем меньше значение фактора любезности, тем выше приоритет процесса. Фактор любезности процесса может быть изменен суперпользователем системным вызовом nice.
* Степень загруженности процессора в момент последнего обслуживания им процесса.

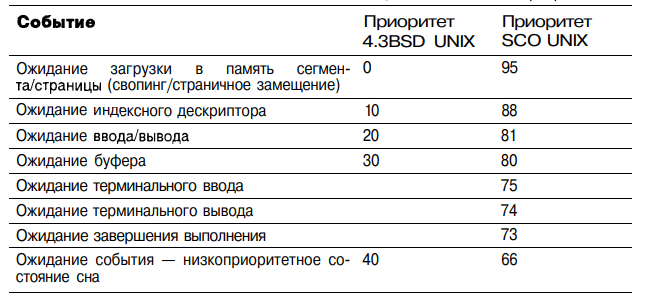
Дескриптор процесса proc содержит следующие поля, относящиеся к приоритету:

* p\_pri – текущий приоритет планирования;
* p\_usrpri – приоритет процесса в режиме задачи;
* p\_cpu – результат последнего измерения степени загруженности процессора процессом;
* p\_nice – фактор любезности.

У процесса, находящегося в режиме задачи, значения p\_pri и p\_usrpri равны. Значение текущего приоритета p\_pri может быть повышено планировщиком для выполнения процесса в режиме ядра.

По пробуждению процесса от блокировки по событию или ресурсу, планировщик устанавливает в поле p\_pri приоритет сна – значение приоритета из диапазона от 0 до 49, зависящее от события или ресурса по которому произошла блокировка.

Системные приоритеты сна приведены в следующей таблице.



При создании процесса поле p\_cpu инициализируется нулем. На каждом тике обработчик таймера увеличивает поле p\_cpu текущего процесса на единицу, до максимального значения, равного 127. Каждую секунду, обработчик прерывания инициализирует отложенный вызов процедуры schedcpy(), которая уменьшает p\_cpu каждого процесса. Процедура schedcpy() пересчитывает приоритеты для режима задачи всех процессов. Увеличение поля p\_cpu приводит к увеличению поля p\_usrpri, следовательно – к снижению приоритета процесса.

**Таким образом, приоритет процесса в режиме задачи может быть динамически пересчитан по следующим причинам:**

* вследствие изменения фактора любезности процесса системным вызовом nice;
* в зависимости от степени загруженности процессора процессом p\_cpu;
* вследствие ожидания процесса в очереди готовых к выполнению процессов;
* приоритет может быть повышен до соответствующего приоритета сна вследствие ожидания ресурса или события.
* процесс может быть переведен из режима задачи в режим ядра, в этом случае его текущий приоритет назначается из диапазона режима ядра.

Динамический пересчет приоритетов процессов в режиме задачи позволяет избежать бесконечного откладывания.

**Пересчет динамических приоритетов в операционных системах семейства Windows.**

Процессу назначается приоритет – базовый приоритет процесса, относительный приоритет потока вычисляется относительно базового приоритет процесса.

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

Планирование осуществляется на основании приоритетов потоков, готовых к выполнению. Поток с более низким приоритетом вытесняется планировщиком, когда поток с более высоким приоритетом становится готовым к выполнению. По истечению кванта времени текущего потока, ресурс передается первому – самому приоритетному - потоку в очереди готовых на выполнения.

Раз в секунду диспетчер настройки баланса - системный поток, предназначенный для выполнения функций управления памятью - сканирует очередь готовых потоков. Если обнаружены потоки, ожидающие выполнения более 4 секунд, диспетчер настройки баланса повышает их приоритет до 15. Как только квант истекает, приоритет потока снижается до базового приоритета процесса путем вычитания всех повышений. Если поток не успел завершить свою работу за квант времени или был вытеснен потоком с более высоким приоритетом, то после снижения приоритета поток возвращается в очередь готовых потоков. Чтобы минимизировать расход процессорного времени, диспетчер настройки баланса сканирует лишь 16 готовых потоков. Кроме того, диспетчер повышает приоритет не более чем у 10 потоков за один проход: обнаружив 10 потоков, приоритет которых следует повысить, он прекращает сканирование. При следующем проходе сканирование возобновляется с того места, где оно было прервано в прошлый раз. Наличие 10 потоков, приоритет которых следует повысить, говорит о необычайно высокой загруженности системы.

Если в системе в текущий момент времени нет потоков, готовых к выполнению, планировщик передает ресурс потоку простоя. Потоки простоя принадлежат процессу простоя.

В Windows используется 32 уровня приоритета: целое число от 0 до 31, где 31 – наивысший приоритет, из них:

* от 16 до 31 – уровни реального времени;
* от 0 до 15 – динамические уровни, уровень 0 зарезервирован для потока обнуления страниц.

При создании процесса, ему назначается базовый приоритет:

* реального времени – Real-time: 24;
* высокий – High: 13;
* выше обычного – Above Normal: 10;
* обычный – Normal: 8;
* ниже обычного – Below Normal: 6;
* уровень простоя – Idle: 4.

Затем потокам внутри процессов назначается относительный приоритет:

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

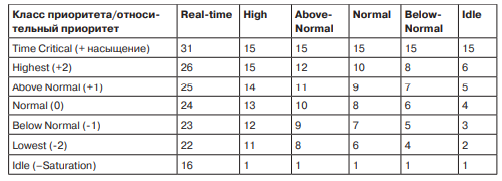
При назначении потоку уровня «критичный по времени» поток получает наивысший возможный для класса его процесса приоритет:

* 31 - для потока процесса уровня реального времени;
* 15 - для потока процесса динамического уровня.

При назначении потоку уровня простоя поток получает самый низкий возможный для класса его процесса приоритет:

* 16 - для потока процесса уровня реального времени;
* 1 - для потока процесса динамического уровня.

В следующей таблице приведены результаты вычисления ядром приоритета потока исходя из базового приоритета процесса и относительного приоритета потока внутри процесса.

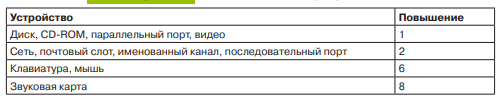


Поток характеризуют два значения приоритета:

* текущее (динамическое);
* базовое.

**Текущий приоритет потока в динамическом диапазоне – от 1 до 15 – может быть повышен планировщиком вследствие следующих причин.**

* Повышение приоритета вследствие событий планировщика или диспетчера.
* Повышение приоритета владельца блокировки.
* Повышение приоритета после завершения ввода/вывода. Значения повышений приведены в следующей таблице.



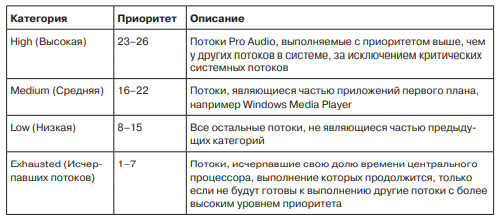
* Повышение приоритета вследствие ввода из пользовательского интерфейса.
* Повышение приоритета вследствие длительного ожидания ресурса исполняющей системы.
* Повышение вследствие ожидания объекта ядра.
* Повышение приоритета в случае, если готовый к выполнению поток не был запущен в течение длительного времени.
* Приоритет мультимедийных потоков повышается службой MMCSS в соответствии с категорией планирования.

Текущий приоритет потока в динамическом диапазоне может быть понижен до базового приоритета путем вычитания всех повышений.

Мультимедийные потоки должны выполняться с минимальными задержками. Эта задача решена в Windows путем повышения приоритетов мультимедийных потоков драйвером MultiMedia Class Scheduler Service (MMCSS). Повышение приоритетов мультимедийных потоков происходит следующим образом. Приложения, которые реализуют воспроизведение мультимедийного контента, указывают драйверу MMCSS задачу из следующего списка:

* аудио;
* захват;
* распределение;
* игры;
* воспроизведение;
* задачи администратора многоэкранного режима.

Для потоков, зарегистрированных с MMCSS определены следующие категории планирования.



Функции MMCSS временно повышают приоритет потоков, зарегистрированных с MMCSS до уровня, соответствующего их категориям планирования. Затем, их приоритет снижается до уровня, соответствующего категории Exhausted, чтобы другие потоки также могли получить ресурс.

**Вывод из проделанной работы.**

Функции обработчика прерывания от системного таймера в защищенном режиме для семейства ОС Windows и для семейства OC Unix/Linux схожи. Обработчики в Windows и в Unix/Linux:

* инициализируют отложенные действия, относящиеся к работе планировщика, такие как пересчет приоритетов;
* выполняют декремент счетчиков времени: часов, таймеров, будильников реального времени, счетчиков времени отложенных действий.
* выполняют декремент кванта: текущего процесса в Linux, текущего потока в Windows;

Несмотря на то, что обе системы являются системами разделения времени с динамическими приоритетами и вытеснением, системы планирования в ОС семейства Windows и семейства Unix/Linux различаются.

* При создании процесса в Windows, ему назначается приоритет, обычно называемый базовым. Приоритеты потоков определяются относительно приоритета процесса, в котором они создаются. Приоритет потока пользовательского процесса может быть пересчитан динамически.
* В Unix/Linux приоритет процесса характеризуется текущим приоритетом и приоритетом процесса в режиме задачи. Приоритет пользовательского процесса - процесса в режиме задачи – может быть динамически пересчитан.