|  |  |
| --- | --- |
| 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»

**Отчёт по лабораторной работе №1 (часть 2) по курсу**

**«Операционные системы»**

**Тема**  Функции обработчика прерываний от системного таймера и пересчет динамических приоритетов

**Студент**  Якуба Д.В.

**Группа**  ИУ7-53Б

**Оценка (баллы)**

**Преподаватель**  Рязанова Н.Ю.

*Москва, 2020 г*

# Функции обработчика прерывания от системного таймера в защищённом режиме

## **Windows**

1. По тику:

* 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;
* Инкремент счётчика системного времени;
* Декремент счётчиков времени отложенных задач;
* При активном механизме профилирования ядра инициализация отложенного вызова обработчика ловушки профилирования ядра путём постановки объекта в очередь DPC (обработчик ловушки профилирования регистрирует адрес команды, выполнявшейся на момент прерывания).

1. По главному тику:

* Освобождения объекта «событие», ожидающего диспетчер настройки баланса.

1. По кванту:

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

## **UNIX/LINUX**

1. По тику:

* Декремент кванта текущего потока;
* Инкремент счетчика тиков аппаратного таймера;
* Декремент счетчиков времени до отправления отложенных задач. Если один из счётчиков достиг нуля, выставление флага для обработчика отложенных вызовов;
* Обновление статистики использования процессора текущим процессом (инкремент поля c\_cpu декриптора текущего процесса до максимального значения 127);
* Обновление системных часов и других таймеров системы.

1. По главному тику:

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

1. По кванту:

* Посылка текущему процессу сигнала SIGXCPU, если тот превысил выделенную ему квоту использования процессора.

# Перерасчёт динамических приоритетов

# Перерасчёт динамических приоритетов в ОС семейства Windows

В операционной системе 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 выполнение.

Раз в секунду диспетчер настройки баланса 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. Если поток не был завершен 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 необычно высокой загруженности системы.

Всего в Windows Всего в Windows 32 уровня запроса прерывания (от 0 до 31). Уровни с 0 по 15 – динамические уровни, где уровень 0 зарезервирован для потока обнуления страниц. Уровни с 16 по 31 – уровни реального времени.

Уровни приоритета потоков назначаются Windwos API и ядром операционной системы. Windows API сортирует процессы по классам приоритета, которые были назначены при их создании:

1. Простой – idle, 1;
2. Обычный – normal, 2;
3. Высокий – high, 3;
4. Реального времени – real-time, 4;
5. Ниже обычного – below normal, 5;
6. Выше обычного – above normal, 6.

После назначается относительный приоритет потоков в рамках процессов:

1. Простой – idle, -15;
2. Низший – lowest, -2;
3. Ниже обычного – below normal, -1;
4. Обычный – normal, 0;
5. Выше обычного – above normal, 1;
6. Высший – highest, 2;
7. Критичный по времени – time critical, 15;

Исходный базовый приоритет потока наследуется от базового приоритета процесса. Процесс по умолчанию наследует свой базовый приоритет у того процесса, который его создал.

Соответствие между приоритетами Windows API и ядра системы приведено в таблицей 1.

Таблица 1, Соответствие между приоритетами Windows API и ядра Windows

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Класс приоритета/относительный приоритет | Real-time | High | Above-normal | Normal | Below-normal | Idle |
| Time Critical | 31 | 15 | 15 | 15 | 15 | 15 |
| Highest | 26 | 15 | 12 | 10 | 8 | 6 |
| Above normal | 25 | 14 | 11 | 9 | 7 | 5 |
| Normal | 24 | 13 | 10 | 8 | 6 | 4 |
| Below normal | 23 | 12 | 9 | 7 | 5 | 3 |
| Lowest | 22 | 11 | 8 | 6 | 4 | 2 |
| Idle | 16 | 1 | 1 | 1 | 1 | 1 |

Текущий приоритет потока в динамическом диапазоне (от 1 до 15) может быть повышен планировщиком вследствие:

* события планировщика или диспетчера;
* повышения приоритета владельца блокировки;
* завершения ввода/вывода (таблица 2)
* ввода из пользовательского интерфейса;
* длительного ожидания ресурса исполняющей системы;
* ожидания объекта ядра;
* случая, когда готовый к выполнению поток не был запущен в течение долгого времени;
* проигрывания мультимедиа службой планировщика MMCSS.

Таблица 2, Рекомендуемые значения повышения приоритета

|  |  |
| --- | --- |
| Устройство | Приращение |
| Диск, CD-ROM, параллельный порт, видео | 1 |
| Сеть, почтовый ящик, именованный канал, последовательный порт | 2 |
| Клавиатура, мышь | 6 |
| Звуковая плата | 8 |

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

Мультимедийные потоки должны выполняться с минимальными задержками. Эта задача решена в Windows Мультимедийные потоки должны выполняться с минимальными задержками. Эта задача решена в Windows путём повышения приоритетов мультимедийных потоков драйвером MMCSS (Multimedia class scheduler service). Повышение приоритетов мультимедийных потоков происходит следующим образом: приложения, которые реализуют воспроизведение мультимедийного контента, указывают драйверу задачу из списка: аудио, захват, распределение, игры, воспроизведение, задачи администратора многоэкранного режима.

Одно из наиболее важных свойств для планирования потоков называется категорией планирования. Это свойство является первичным фактором, определяющим приоритет потоков, зарегистрированных с MMCSS. В следующей таблице предоставлены категории планирования.

Таблица 3, Категории планирования

|  |  |  |
| --- | --- | --- |
| Категория | Приоритет | Описания |
| Exhausted | 1-7 | Потоки, исчерпавшие свою долю времени центрального процессора, выполнение которых продолжиться, только если не будут готовы к выполнению другие потоки с более высоким уровнем приоритета |
| Low | 8-15 | Все остальные потоки, не являющиеся частью категорий Medium, High |
| Medium | 16-22 | Потоки, являющие частью приложений первого плана, например, Windows Media Player |
| High | 23-26 | Потоки профессионального аудио (Pro Audio), запущенные с приоритетом выше, чем у других потоков на системе, за исключением критических системных потоков |

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 доступ к требуемому ресурсу.

# Перерасчёт динамических приоритетов в ОС семейства UNIX/LINUX

В современных системах UNIX/Linux 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.

Очередь готовых к выполнению процессов формируется согласно приоритетам процессов и принципу вытесняющего циклического планирования: в первую очередь выполняются процессы с большим приоритетом, а 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.

Приоритет представляет собой целое число из диапазона от 0 до 127:

* От 0 до 49 – приоритеты ядра;
* От 50 до 127 – приоритеты прикладных задач.

Приоритеты ядра являются фиксированными величинами.

Приоритеты прикладных задач могут изменяться во времени в зависимости от фактора любезности и последней измеренной величины использования процессора.

Фактор любезности является целым числом от 0 до 39. 20 – значение фактора по умолчанию. Чем меньше значение фактора любезности, тем выше приоритет процесса. Фоновым процессам автоматически задаются более высокие значения фактора. Фактор любезности процесса может быть изменен суперпользователем с помощью системного вызова nice.

Дескриптор процесса proc содержит следующие поля приоритета:

1. p\_pri – текущий приоритет планирования;
2. p\_usrpri – приоритет процесса в режиме задачи;
3. p\_cpu – результат последнего измерения степени загруженности процессора процессом;
4. 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 планироващиком для выполнения процесса в режиме ядра (при этом p\_usrpri будет использоваться для хранения приоритета, который будет назначен при возврате в режим задачи).

Ядро системы связывает приоритет сна с событием или ожидаемым ресурсом, из-за которого процесс может блокироваться. Когда процесс просыпается после блокирования в системном вызове, ядро устанавливает в поле p\_pri приоритет сна – значение проиоритета из диапазона от 0 до 49, зависящее от события или ресурса, по которому произошла блокировка. Событие и связанное с ним значение приоритета сна в системе 4.3BSD описывает таблица 4.

Таблица 4, Таблица приоритетов в системе 4.3BSD

|  |  |  |
| --- | --- | --- |
| Приоритет | Значение | Описание |
| PSWP | 0 | Свопинг |
| PSWP + 1 | 1 | Страничный демон |
| PSWP + 1/2/4 | 1/2/4 | Другие действия по обработке памяти |
| PINOD | 10 | Ожидание освобождения inode |
| PRIBIO | 20 | Ожидание дискового ввода-вывода |
| PRIBIO + 1 | 21 | Ожидание освобождения буфера |
| PZERO | 25 | Базовый приоритет |
| TTIPRI | 28 | Ожидание ввода с терминала |
| TTOPRI | 29 | Ожидание вывода с терминала |

При создании процесса поле c\_cpu иницализируется нулём. 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 значение p\_cpu каждого процесса, исходя из фактора «полураспада».

В системе 4.3BSD для расчёта фактора полураспада применяется формула:

– среднее количество процессов, находящихся в состоянии готовности к выполнению за последнюю секунду.

Процедура schedcpy() пересчитывает приоритеты для режима задачи всех процессов по формуле:

– базовый приоритет в режиме задачи, равный 50.

В результате, если процесс в последний раз использовал большое количество процессорного времени, его p\_cpu будет увеличен. Это приведёт к росту значения p\_usrpri и, следовательно, к понижению приоритета. Чем дольше процесс простаивает в очереди на исполнение, тем больше фактор полураспада уменьшает его p\_cpu, что приводит к повышению его приоритета. Такая схема предотвращает зависание низкоприоритетных процессов по вине операционной системы. Её применение предпочтительнее процесса, осуществляющим много операций ввода-вывода, в противоположность процессам, производящим много вычислений.

# Вывод