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 по лабораторной работе по курсу "Операционные системы"

Тема Прерывания таймера в Windows и Unix

Студент Кузнецова А. В.

Группа ИУ7-51Б

Преподаватель Рязанова Н. Ю.

Москва — 2022 г.

Содержание

1 Функции обработчика прерывания от системного таймера

1.1 ОС семейства Unix

1.1.1 По тику

В задачи обработчика прерывания от системного таймера по тику входит:

- инкремент счетчика времени с момента запуска системы (SVR4);
- инкремент счетчика реального времени;
- декремент кванта;
- декремент счетчиков времени и при достижении счетчиками нуля установка флага для обработчиков отложенных вызовов;
- инкремент счетчика процессорного времени, полученного процессом в режиме задачи и в режиме ядра.

1.1.2 По главному тику

В задачи обработчика прерывания от системного таймера по главному тику входит:

- инициализация отложенного вызова функции планировщика;
- инициализация отложенного вызова процедуры *wakeup*, которая меняет состояние процесса с «спящего» на «готовый к выполнению»;
- пробуждение системных процессов, таких как *swapper* и *pagedaemon*;
- декремент счетчика времени, которое осталось до отправления одного из следующих сигналов:

– *SIGALRM* - сигнал, посылаемый процессу по истечении промежутка реального времени;

– *SIGPROF* - сигнал, посылаемый процессу по истечении времени, которое задано в таймере профилирования;

– *SIGVTALRM* - сигнал, посылаемый процессу по истечении времени, которое задано в ”виртуальном” таймере.

1.1.3 По кванту

Обработчик прерывания от системного таймера по кванту посылает сигнал *SIGXCPU* текущему процессу, если он превысил выделенную для него квоту процессорного времени.

1.2 ОС семейства Windows

1.2.1 По тику

В задачи обработчика прерывания от системного таймера по тику входит:

- инкремент счетчика реального времени;
- декремент кванта текущего потока;
- декремент счетчиков отложенных задач.

1.2.2 По главному тику

Обработчик прерывания от системного таймера по главному тику инициализирует диспетчер настройки баланса, освобождая объект "событие", на котором он ожидает.

1.2.3 По кванту

Обработчик прерывания от системного таймера по кванту инициализирует диспетчеризацию потоков путем постановки соответствующего объекта в очередь DPC.

2 Пересчет динамических приоритетов

Системы семейств Unix и Windows являются системами разделения времени с динамическими приоритетами и вытеснением. Динамические приоритеты могут иметь только пользовательские процессы.

2.1 ОС семейства Unix

При создании процесса ему назначается базовый приоритет. Очередь процессов, готовых к выполнению, формируется согласно приоритетам процессов и принципу вытесняющего циклического планирования. В первую очередь выполняются процессы, имеющие больший приоритет. Процессы с одинаковыми приоритетами выполняются циклически друг за другом, в течение кванта времени. В случае, если процесс с более высоким приоритетом поступает в очередь готовых к выполнению процессов, планировщик вытесняет текущий процесс и предоставляет ресурс более приоритетному процессу.

Приоритет задается целым числом из диапазона от 0 до 127. Чем меньше число, тем выше приоритет процесса. Приоритеты от 0 до 49 зарезервированы ядром опреационной системы, в связи с чем пользовательские процессы могут обладать приоритетом от 50 до 127.

В отличие от приоритетов ядра, являющихся фиксированными величинами, приоритеты пользовательских процессов могут изменяться во времени в зависимости от двух факторов:

- фактор любезности. Чем меньше его значение, тем выше приоритет процесса. Фактор любезности может быть изменен с помощью системного вызова *nice*, однако только суперпользователь может увеличивать значение приоритета;
- фактор использования, который определяется степенью последней загруженности CPU процессом.

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.

Планировщик использует значение *p_pri* для принятия решения о том, какой процесс направить на выполнение. У процесса, который находится 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 (*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 блокировки, ядро устанавливает в поле *p_pri* приоритет сна — значение приоритета из диапазона от 0 до 49, зависящее от события или ресурса по которому произошла блокировка. Значения приоритетов сна для событий в системе *4.3BSD* описаны в таблице 2.1.

Таблица 2.1 – Приоритеты сна в 4.3BSD

Приоритет	Значение	Описание
PSWP	0	Свопинг
PSWP + 1	1	Страничный демон
PSWP + 1/2/4	1/2/4	Другие действия при 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
TTORPI	29	Ожидание вывода с терминала
PWAIT	30	Ожидание завершения процесса потомка
PLOCK	35	Консультативное ожидание блокированного ресурса
PSLEP	40	Ожидание сигнала

При инициализации процесса поле p_cpu равно нулю. На каждом тике обработчик прерывания увеличивает поле p_cpu текущего процесса на единицу, до максимального значения, равного 127. Каждую секунду, обработчик прерывания инициализирует отложенный вызов процедуры *schedcpu()*, которая уменьшает значение p_cpu каждого процесса исходя из фактора "полураспада".

Для расчёта фактора полураспада применяется формула (2.1).

$$decay = \frac{2 \cdot load\_average}{2 \cdot load\_average + 1} \tag{2.1}$$

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

Процедура *schedcpu()* пересчитывает приоритеты для режима задачи всех процессов по формуле (2.2).

$$p\_usrpri = PUSER + \frac{p\_cpu}{2} + 2 \cdot p\_nice \tag{2.2}$$

где $PUSER$ - базовый приоритет в режиме задачи, равный 50.

В результате *p_cpu* будет увеличен, если процесс до вытеснения другим процессом использовал большое количество процессорного времени. Это приведет к росту значения *p_usrpri* и, следовательно, к понижению приоритета. Чем дольше процесс простаивает в очереди на выполнение, тем больше фактор полураспада уменьшает его *p_cpu*. Такая схема предотвращает бесконечное откладывание низкоприоритетных процессов в ОС.

2.2 ОС семейства Windows

В Windows процессу при создании назначается базовый приоритет. В Windows реализовано вытесняющее планирование на основе уровней приоритета: если поток с более высоким приоритетом становится готовым к выполнению, поток с более низким приоритетом вытесняется планировщиком, даже если квант текущего не истек. По истечению кванта времени текущего потока, ресурс передается самому приоритетному потоку в очереди готовых к выполнению.

В Windows *диспетчер настройки баланса* раз в секунду сканирует очередь готовых потоков. Если обнаружены потоки, ожидающие выполнения более 4 секунд, диспетчер настройки баланса повышает их приоритет до 15. Когда истекает квант, 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.

Для того, чтобы минимизировать расход процессорного времени, диспетчер настройки баланса сканирует только 16 попозиций в очереди и повышает приоритет не более чем у 10 потоков за один проход. При обнаружении 10 потоков, приоритет которых следует повысить, диспетчер настройки баланса прекращает сканирование. При следующем проходе сканирование возобновляется с того места, где оно было прервано.

В Windows используется 32 уровня приоритета:

- от 0 до 15 - динамические уровни (уровень 0 зарезервирован для потока обнуления страниц);

- от 16 до 31 - приоритеты процессов реального времени (31 — наивысший).

Уровни приоритета потоков назначаются с дыух позиций: Windows API и ядра ОС.

Сначала Windows API сортирует процессы по классу приоритета, которые были назначены при их создании:

- реального времени (Real-time) - (4);
- высокий (High) - (3);
- выше обычного (Above Normal) - (6);
- обычный (Normal) - (2);
- ниже обычного (Below Normal) - (5);
- простой (Idle) - (1).

Затем назначается относительный приоритет потоков процесса:

- критичный по времени (Time-critical) - (15);
- наивысший (Highest) - (2);
- выше обычного (Above-normal) - (1);
- обычный (Normal) - (0);
- ниже обычного (Below-normal) - (–1);
- низший (Lowest) - (–2);
- простой (Idle) - (–15)

В таблице 2.2 показано соответствие между приоритетами Windows API и ядра системы.

Таблица 2.2 – Соответствие между приоритетами Windows API и ядра Windows

Класс приоритета	Real-time	High	Above	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

Планировщик может повысить текущий приоритет потока в динамическом диапазоне (от 1 до 15) по следующим причинам:

- повышение вследствие событие планировщика или диспетчера;
- повышение приоритета владельца блокировки;
- завершение операций ввода/вывода (таблица 2.3);

Таблица 2.3 – 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

- ввод из пользовательского интерфейса;
- длительное ожидание ресурса исполняющей системы;
- ожидание объекта ядра;

- готовый к выполнению поток не был запущен в течение длительного времени;
- повышение приоритета службой планировщика MMCSS.

2.2.1 MMCSS

Для того, чтобы потоки, на которых выполняются различные мультимедийные приложения могли выполняться с минимальными задержками, драйвур *MMCSS* (*MultiMedia Class Scheduler Service*) временно повышает приоритет таких потоков.

MMCSS работает со следующими задачами:

- звук;
- возможность использования функции записи;
- воспроизведение звукового или видео контента;
- аудио профессионального качества;
- задачи администратора многооконного режима.

Одно из наиболее важных свойств для планирования потоков — категория планирования (Scheduling Category) — первичный фактор определяющий приоритет потоков, зарегестрированных в MMCSS (таблица 2.4).

Таблица 2.4 – Категории планирования

Категория	**Приоритет**	**Описание**
High (Высокая)	23-26	Потоки профессионального аудио (Pro Audio), запущенные с приоритетом выше, чем у других потоков на системе, за исключением критических системных потоков
Medium (Средняя)	16-22	Потоки, являющиеся частью приложений первого плана, например Windows Media Player
Low (Низкая)	8-15	Все остальные потоки, не являющиеся частью предыдущих категорий
Exhausted (Исчерпавших потоков)	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

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

2.2.2 Уровни запросов прерываний

Хотя контроллеры прерываний устанавливают приоритетность прерываний, Windows устанавливает свою собственную схему приоритетности прерываний, известную как уровни запросов прерываний (IRQL). В ядре IRQL-уровни представлены в виде номеров от 0 до 31 на системах x86, где более высоким номерам соответствуют прерывания с более высоким приоритетом.

На рис. 2.1 показаны IRQL-уровни для архитектуры x86.

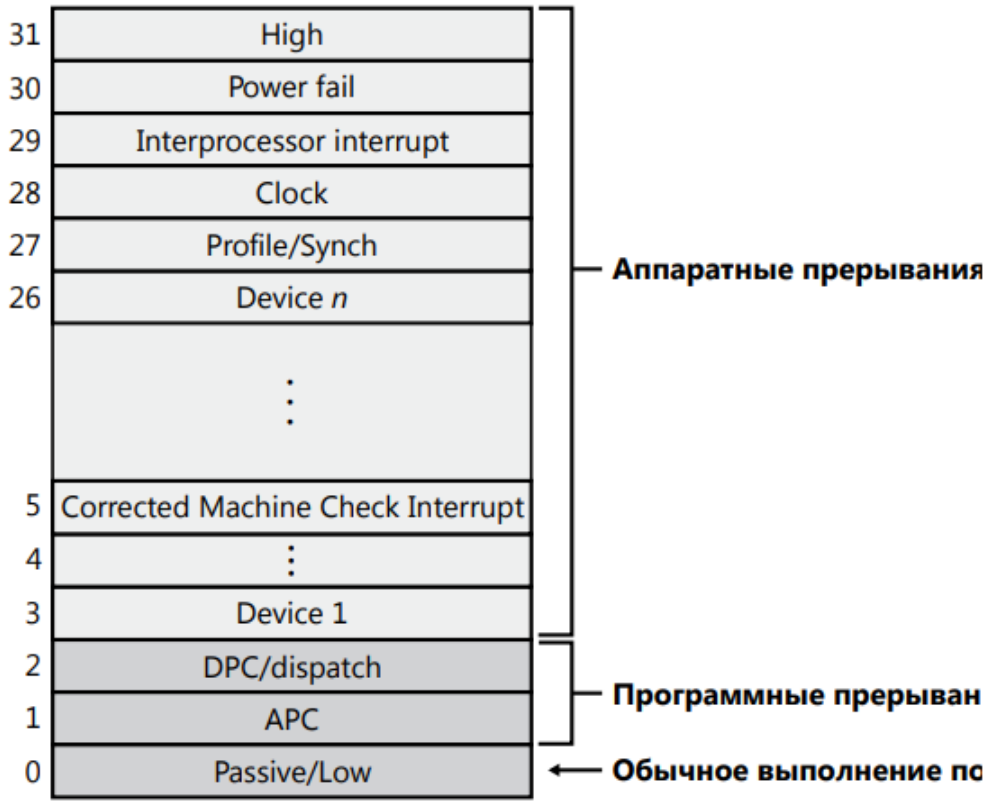

Рисунок 2.1 – Уровни запросов прерываний

Прерывания обслуживаются в порядке их приоритета, и прерывания с более высоким уровнем приоритета получают преимущество в обслуживании. При возникновении прерывания с высоким уровнем приоритета процессор сохраняет состояние прерванного потока и запускает связанный с прерыванием диспетчер системных прерываний.

Вывод

Обработчик прерывания от системного таймера для ОС семейства Windows и для ОС семейства Unix выполняет следующие схожие функции:

- декремент счетчиков времени (часов, таймеров, счетчиков времени отложенных действий, будильников реального времени);
- декремент кванта;
- инициализация отложенных действий, которые относятся к работе планировщика, например, пересчет приоритетов.

Данная схожесть обусловлена тем, что ОС Unix и ОС Windows — системы разделения времени с динамическими приоритетами и вытеснением.

Пересчет динамических приоритетов осуществляется только для пользовательских процессов для того, чтобы избежать бесконечного откладывания.