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 № 1 часть 2

по курсу «Операционные системы»

на тему: «Прерывание таймера в Windows и Unix»

Студент ИУ7-53Б _____ Д. В. Недолужко
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Н. Ю. Рязанова
(Подпись, дата) (И. О. Фамилия)

2021 г.

1 Функции обработчика прерывания от системного таймера

Тик — период времени между двумя последующими прерываниями таймера.

Главный тик — период времени между N последующими прерываниями таймера. Число N зависит от конкретного варианта системы.

Квант времени — временной интервал, в течение которого процесс может использовать процессор до вытеснения другим процессом.

1.1 Unix/Linux

Обработчик прерывания от системного таймера выполняет следующие функции

1.1.1 По тику

- Ведет счёт тиков аппаратного таймера по необходимости
- Обновляет статистику использования процессора текущим процессом. Происходит инкремент поля p_cpu структуры proc текущего процесса. Данное поле при создании процесса инициализируется нулем, а увеличивается до максимального значения, равного 127.
- Инкремент часов и других таймеров системы. В SVR4 происходит инкремент переменной lbolt, хранящей количество тиков, отсчитанных с момента загрузки системы.
- Декремент кванта

1.1.2 По главному тику

- Пробуждает в нужные моменты системные процессы, такие как swapper и pagedaemon. Под пробуждением понимается регистрация отложенного вызова процедуры wakeup, которая перемещает дескрипторы процессов из списка “спящих” в очередь готовых к выполнению.
- Декрементирует счётчик времени, оставшегося времени до посылки одного из следующих сигналов:

- $SIGALRM$ – сигнал, посылаемый процессу по истечении времени, предварительно заданного функцией alarm();
- $SIGPROF$ – сигнал, посылаемый процессу по истечении времени заданного в таймере профилирования;
- $SIGVTALRM$ – сигнал, посылаемый процессу по истечении времени, заданного в виртуальном таймере.

1.1.3 По кванту

- Посылает текущему процессу сигнал $SIGXCPU$, если тот превысил выделенную ему квоту использования процессора

1.2 Windows

Обработчик прерывания от системного таймера выполняет следующие функции

1.2.1 По тику

- Инкремент счетчика системного времени
- Декремент счетчика времени отложенные задач
- 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)
- если активен механизм профилирования ядра, то инициализирует отложенный вызов обработчика ловушки профилирования ядра путём постановки объекта в очередь DPC(Deferred procedure call (отложенный вызов процедуры)): обработчик ловушки профилирования регистрирует адрес команды, выполнявшейся на момент прерывания.

1.2.2 По главному тику

- Освобождает объект «событие», который ожидает диспетчер настройки баланса.

1.2.3 По кванту

- Инициализация диспечерезации потоков путем постановки соответствующего объекта DPC в очередь

2 Пересчет динамических приоритетов

2.1 Unix/Linux

2.1.1 Приоритеты процессов

Приоритет процесса задается любым целым числом, лежащим в диапазоне от 0 до 127. Чем меньше такое число, тем выше приоритет. Приоритеты от 0 до 49 зарезервированы для ядра, следовательно, прикладные процессы могут обладать приоритетом в диапазоне 50-127. Структура proc содержит следующие поля, относящиеся к приоритетам:

- p_pri – Текущий приоритет планирования
- p_usrpri – Приоритет режима задачи
- p_cpu – Результат последнего измерения использования процессора
- p_nice – Фактор «любезности», устанавливаемый пользователем

Планировщик использует p_pri для принятия решения о том, какой процесс направить на выполнение. Когда процесс находится в режиме задачи, значение его p_pri идентично p_usrpri. Когда процесс просыпается после блокирования в системном вызове, его приоритет будет временно повышен для того, чтобы дать ему предпочтение для выполнения в режиме ядра. Следовательно, планировщик использует p_usrpri для хранения приоритета, который будет назначен процессу при возврате в режим задачи, а p_pri — для хранения временного приоритета для выполнения в режиме ядра.

Приоритет в режиме задачи зависит от двух факторов: «любезности» (nice) и последней измеренной величины использования процессора. Степень любезности (nice) является числом в диапазоне от 0 до 39 со значением 20 по умолчанию. Увеличение значения приводит к уменьшению приоритета. Фоновым процессам автоматически задаются более высокие значения степени благоприятствия. Уменьшить эту величину для какого-либо процесса может только суперпользователь, поскольку при этом поднимется его приоритет.

Поле p_cpu структуры proc содержит величину результата последнего сделанного измерения использования процессора процессом. При создании процесса значение этого поля инициализируется нулем. На каждом тике

обработчик таймера увеличивает p_cpu на единицу для текущего процесса до максимального значения, равного 127.

В современных UNIX ядро является полностью вытесняемым. Например, *Solaris2.x*, *Linux*.

2.1.2 Перерасчет приоритетов

Каждую секунду ядро системы вызывает процедуру schedcpu() (запускаемую через отложенный вызов), которая уменьшает значение p_cpu каждого процесса исходя из фактора «полураспада» (delay factor). В системе SVR3 используется фиксированное значение этого фактора, равное $\frac{1}{2}$.

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

$$p\_usrpri = PUSER + \frac{p\_cpu}{4} + 2 \cdot p\_nice$$

где PUSER — базовый приоритет в режиме задачи, равный 50.

Процедура schedcpu() пересчитывает приоритет каждого процесса каждую секунду. Так как приоритет процесса, находящегося в очереди на выполнение, не может быть изменен, процедура schedcpu() извлекает процесс из очереди, меняет его приоритет и помещает его обратно. Обработчик прерываний таймера пересчитывает приоритет текущего процесса через каждые четыре тика.

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 приоритета из диапазона от 0 до 49, зависящее от события или ресурса по которому произошла блокировка. На рисунке 2.1 показано событие и связанное с ним значение приоритета сна в системе 4.3BSD.

2.1.3 Переключение контекста

Существуют три ситуации, при которых возникает переключение контекста.

- Если текущий процесс блокируется в ожидании ресурса или завершает

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

Рисунок 2.1 – Системы приоритета сна 4.3BSD

работу. При этом происходит свободное переключение контекста.

- Если в результате, полученном процедурой пересчета приоритетов, оказалось, что другой процесс обладает более высоким приоритетом по сравнению с текущим.

- Если текущий процесс или обработчик прерываний разбудил более приоритетный процесс.

2.2 Windows

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 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

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

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) и 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) и Простоя — Idle (–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 на таблице 2.1.

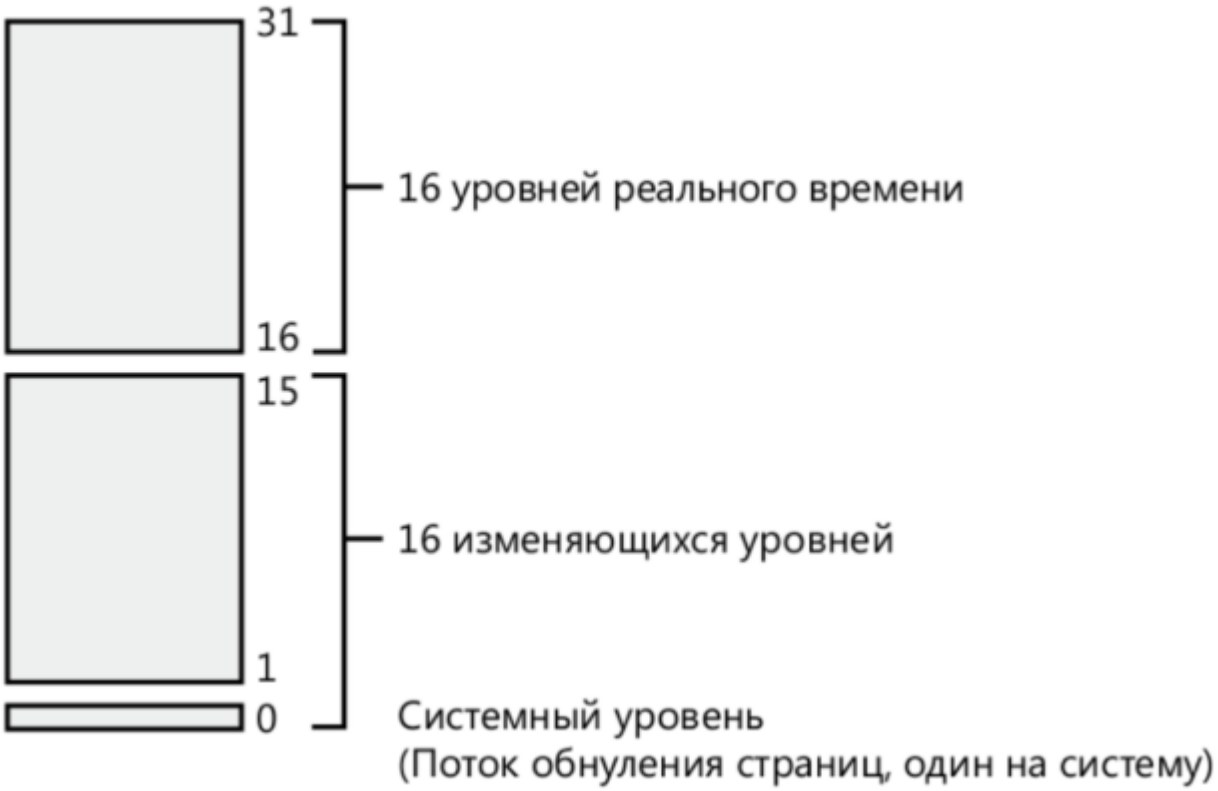

Рисунок 2.2 – Уровни приоритета потоков

Текущий приоритет потока в динамическом диапазоне — от 1 до 15 — может быть повышен планировщиком вследствие следующих причин:

- повышение вследствие событие планировщика или диспетчера(сокращение задержек);
- повышение приоритета владельца блокировки;
- повышение приоритета после завершения ввода/вывода (сокращение задержек) (таблица 2.2);
- повышение приоритета вследствие ввода из пользовательского интерфейса(сокращение задержек и времени отклика);
- повышение приоритета вследствие длительного ожидания ресурса исполняющей системы(предотвращение зависания);
- повышение вследствие ожидания объекта ядра;

Таблица 2.1 – Соответствие между приоритетами Windows API и ядра Windows

	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

- повышение приоритета в случае, когда готовый к выполнению поток не был запущен в течение длительного времени (предотвращение зависания и смены приоритетов);
- повышение приоритета проигрывания мультимедиа службой планировщика MMCSS

Таблица 2.2 – Рекомендуемые значения повышения приоритета

Устройство	Повышение приоритета
Жесткий диск, привод компакт-дисков, параллельный порт, видео устройство	1
Сеть, почтовый порт, именованный канал, последовательный порт	2
Клавиатура, мышь	6
Звуковое устройство	8

2.2.1 MMCSS

Повышение приоритета проигрывания мультимедиа обычно управляется службой пользовательского режима, которая называется службой планировщика класса мультимедиа — MultiMedia Class Scheduler Service (MMCSS). MMCSS работает с вполне определенными задачами, включая следующие

- аудио;
- игры;
- построчное воспроизведение текущего изображения в видеобуффер;
- воспроизведение медиа-контента;
- задачи администратора многоэкранного режима.

В свою очередь, каждая из этих задач включает информацию о свойствах, отличающих их друг от друга. Одно из наиболее важных свойств для планирования потоков называется категорией планирования — Scheduling Category, которое является первичным фактором, определяющим приоритет потоков, зарегистрированных с MMCSS. На таблице 2.3 показаны различные категории планирования.

Механизм, положенный в основу MMCSS, повышает приоритет потоков внутри зарегистрированного процесса до уровня, соответствующего их категории планирования.

Затем он снижает категорию этих потоков до Exhausted, чтобы другие, не относящиеся к мультимедийным приложениям потоки, также могли получить ресурс

Таблица 2.3 – 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

Вывод

Функции обработчика прерывания от системного таймера в защищенном режиме в системах Unix и Windows выполняют одинаковые действия:

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

Обе системы являются системами разделения времени с динамическими приоритетами и вытеснением.

Приоритеты пользовательских процессов является динамическим, а приоритет ядра является фиксированной величиной.

При пересчете приоритетов пользовательских процессов учитывается время ожидания для исключения бесконечного откладывания.

Повышение приоритетов после ввода с устройств интерактивного ввода обусловлено сокращение задержек для более быстрой работы системы.