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: «Обработчик прерывания системного таймера и пересчет динамических приоритетов»

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

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

2023 г.

СОДЕРЖАНИЕ

1 Функции обработчика прерывания системного таймера в системах разделения времени

1.1 Функции обработчика прерывания системного таймера в семействе ОС Windows

По кванту

Обработчик прерывания системного таймера ставит DPC-вызов в очередь, чтобы инициировать диспетчеризацию потоков, а затем завершить свою работу и понизить IROL-уровень процессора.

По тику

– декремент кванта;

– инкремент счетчика реального времени;

– декремент счетчиков времени до выполнения отложенных задач.

По главному тику

– освобождение объекта «событие», которое ожидает диспетчер настройки баланса. Диспетчер настройки баланса сканирует очередь готовых процессов и повышает приоритет процессов, которые находились в состоянии ожидания дольше 4 секунд;

– постановка в очередь DPC отложенного вызова обработчика ловушки профилирования ядра.

1.2 Функции обработчика прерывания системного таймера в семействе ОС UNIX/Linux

По кванту

– посылка текущему процессу сигнала SIGXCPU, если он превысил выделенный для него квант процессорного времени.

По тику

– декремент кванта;

– инкремент счётчика тиков;

– обновление статистики использования процессора текущим процессом;

– декремент счетчиков отложенных действий и их инициализация.

По главному тику

– инициализация отложенных действий;

– инициализация отложенного вызова процедуры *wake_up*, которая перемещает дескрипторы процессов из очереди «спящих» в очередь готовых к выполнению;

– декремент счетчика времени до отправки дного из сигналов:

- SIGALRM (посылается процессу после истечения времени, установленного с помощью функции alarm());
- SIGPROF (посылается процессу после истечения времени, установленного в таймере профилирования);
- SIGVTALRM (посылается процессу после истечения времени, установленного в «виртуальном таймере»).

2 Пересчет динамических приоритетов

Приоритет процесса зависит от количества используемого им процессорного времени и от того, на каком объекте ядра процесс был блокирован.

2.1 Пересчет динамических приоритетов в семействе ОС Windows

При создании процессам назначается приоритет планирования. Приоритеты потоков являются относительными к приоритетам процессов.

В Windows реализуется приоритетная, вытесняющая система планирования, при которой всегда выполняется хотя бы один работоспособный (готовый) поток с самым высоким приоритетом.

Windows использует 32 уровня приоритета:

- 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 (6);
- 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).

Соответствие между Windows-приоритетами и внутренними номерными приоритетами Windows показаны в таблице 2.1.

Таблица 2.1 – Отображение приоритетов ядра Windows на Windows API

Класс приоритета/ Относ-й приоритет	**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

Повысить или понизить приоритет потока в динамическом диапазоне можно в любом приложении, но для этого нужны привилегии, позволяющие повышать приоритет, использующийся при планировании для ввода значения в пределах динамического диапазона.

Повышение приоритета

Планировщик Windows периодически изменяет текущий приоритет потоков, используя внутренний механизм повышения приоритета.

Повышение приоритета вследствие событий планировщика или диспетчера

При наступлении события диспетчера, происходит вызов процедуры для обработки списка готовых потоков, отложенных на выполнение, при этом вызывающий код может указать какого типа повышение должно быть применено к потоку, а также с каким приращением приоритета должно быть связано это повышение.

Повышения приоритета, связанные с завершением ожидания

При пробуждении потока и перемещении его в очередь готовых к исполнению, его приоритет должен быть повышен, чтобы поток как можно скорее начал выполняться.

Повышение приоритета после завершения ввода-вывода

Windows дает временное повышение приоритета при завершении определенных операций ввода/вывода, при этом потоки, ожидавшие ввода/вывода, имеют больше шансов сразу же запуститься и обработать то, чего они ожидали. В таблице 2.2 перечислены рекомендуемые значение повышения приоритета.

Таблица 2.2 – Рекомендуемые значение 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

Повышение при ожидании ресурсов исполняющей системы

Когда поток пытается получить ресурс, уже находящийся в исключительном владении, он переходит в режим ожидания, пока ресурс не будет освобожден. Система ограничивает длительность этого ожидания, выполняя его интервально каждые 500 мс. Если ресурс остается недоступным даже после этого времени, система предотвращает блокировку процессора путем изменения приоритета потоков и выполнения дополнительных ожиданий.

Повышение приоритета потоков первого плана после ожидания

Предоставление небольшого повышения приоритета приложениям первого плана по окончании ожидания способствует более оперативному началу их работы, особенно в ситуациях, когда другие процессы с тем же базовым приоритетом могут быть запущены в фоновом режиме.

Повышение приоритета после пробуждения GUI-потока

Потоки — владельцы окон получают при пробуждении дополнительное повышение приоритета на 2 из-за активности при работе с окнами, например, при поступлении сообщений от окна.

Повышения приоритета, связанные с перезагруженностью центрального процессора (CPU Starvation)

Одна из частей потока, которую называют диспетчером настройки баланса (ДНБ), несёт функцию ослабления загруженности центрального процессора. ДНБ сканирует очередь готовых потоков раз в секунду и, если обнаружены потоки, ожидающие выполнения более 4 секунд, то 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.

Повышения приоритетов для мультимедийных приложений и игр

Для решения проблемы вытеснения потоков, на которых выполняются мутимедийные приложения и игры, в Windows используют драйвер MMCSS (MultiMedia Class Scheduler Service). Клиентские приложения регистрируются у MMCSS, передавая имя задачи, например, аудио, игры, воспроизведение и др. Одно из наиболее важных свойств для планирования потоков называется категорией планирования (Scheduling Category) — это основной фактор, определяющий приоритет потоков, зарегистрированных с MMCSS. В таблице 2.3 показаны различные категории планирования.

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 продолжится, только если не будут готовы к выполнению другие потоки с более высоким уровнем приоритета

2.2 Пересчет динамических приоритетов в семействе ОС UNIX/Linux

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, которые относятся к приоритетам:

- p_runpri — приоритет выполнения процесса в текущий момент времени;
- p_slppri — приоритет, который используется для упорядочения процессов в состоянии ожидания;
- p_usrpri — приоритет режима задачи;
- p_estcpu — результат последнего измерения использования процессора;
- p_nice — фактор «любезности», который устанавливается пользователем.

Планировщик использует p_runpri для принятия решения о том, какой процесс направить на выполнение, а именно для хранения временного приоритета для выполнения в режиме ядра.

Поле p_usrpri используется для хранения приоритета, который будет назначен процессу при возврате в режим задачи из состояния блокировки.

У событий или объектов ядра, на которых может быть блокирован процесс, определён приоритет сна. Приоритет сна являтся величиной, определяемой для ядра, и потому лежит в диапазоне 0-49.

Таблица 2.4 – Системные приоритеты сна

4.3BSD UNIX	**SCO UNIX**	**Событие**
0	95	Ожидание загрузки в память сегмента/страницы (свопинг, страничное замещение)
10	88	Ожидание индексного дескриптора
20	81	Ожидание ввода-вывода
30	80	Ожидание буфера
-	75	Ожидание терминального ввода
-	74	Ожидание терминального вывода
-	73	Ожидание завершения выполнения
40	66	Ожидание события — низкоприоритетное состояние сна

Таблица 2.5 – Приоритеты сна в ОС **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
PWAIT	30	Ожидание завершения процесса потомка
PLOCK	35	Консультативное ожидание блок. ресурса
PSLEP	40	Ожидание сигнала

Можно выделить следующие особые ситуации, связанные с изменением полей p_usrpri и p_runpri:

– когда процесс находится в режиме задачи, то его значения полей p_usrpri и p_runpri равны;

– когда процесс просыпается после блокирования, то есть происходит его постановка в очередь готовых процессов, его приоритету p_runpri присваивается значение приоритета сна события или ресурса, на котором он был блокирован, чтобы дать процессу предпочтение для выполнения в режиме ядра;

– когда процесс завершил выполнение системного вызова и находится в состоянии возврата в режим задачи, его приоритет p_runpri сбрасывается обратно в значение текущего приоритета в режиме задачи p_usrpri.

Приоритет в режиме задачи зависит от двух факторов: «любезности» (p_nice) и последней измеренной величины использования процессора (p_estcpu).

Степень любезности ($nice\ value$) является числом в диапазоне от 0 до 39 со значением 20 по умолчанию. Увеличение значения приводит к уменьшению приоритета.

Каждую секунду ядро системы инициализирует отложенный вызов процедуры $schedcpu()$, которая уменьшает значение p_runpri каждого процесса исходя из фактора «полураспада» ($decay\ factor$). В системе SVR3 используется фиксированное значение этого фактора, равное $\frac{1}{2}$.

В 4.3BSD фактор полураспада рассчитывается по формуле 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.

Также процедура *schedcpu*() пересчитывает приоритеты для режима задачи всех процессов по формуле 2.2,

$$p\_usrpri = PUSER + \frac{p\_estcpu}{4} + 2 \cdot p\_nice \tag{2.2}$$

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

Таким образом, если процесс в последний раз использовал большое количество процессорного времени, то его $\_estu$ будет увеличен. Исходя из формулы 2.2, значение p_estcpu увеличится, соответственно приоритет процесса понизится. Чем дольше процесс простаивает в очереди на выполнение, тем больше фактор полураспада уменьшает его $\_estu$, что приводит к повышению его приоритета.

Такая схема предотвращает бесконечное откладывание процессов. Применение данной схемы предпочтительно процессам, осуществляющим много операций ввода-вывода, в противоположность процессам, производящим много вычислений. То есть динамический пересчет приоритетов процессов в режиме задачи позволяет избежать бесконечного откладывания.

Вывод

Функции обработчика прерывания от системного таймера в операционных системах семейства ОС UNIX/Linux и семейства ОС Windows выполняют схожие задачи, поскольку системы обоих семейств являются являются системами разделения времени с вытеснением и динамически управляемыми приоритетами:

- декремент кванта;
- декремент счетчиков отложенных действий и их инициализация;
- добавление функций планировщика в очередь отложенных действий;
- инкремент счетчика реального времени.