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: «Функции обработчика прерывания от системного таймера и пересчет динамических приоритетов в системах разделения времени»

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

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

2023 г.

СОДЕРЖАНИЕ

1 Функции обработчика прерывания от системного таймера в системах разделения времени

1.1 UNIX/Linux

Обработчик прерывания от системного таймера выполняет несколько функций.

По **тику**:

— инкремент счетчика реального времени;

— декремент кванта текущего потока;

— обновление статистики использования потока текущим процессом — инкремент поля **c_cpu** дескриптора текущего потока до максимального значения 127;

— декремент счетчиков времени до отправки на выполнение отложенных вызовов, при достижении счетчиком нуля происходит установка флага для обработчика отложенных вызовов.

По **главному тику**:

— инициализацию отложенных действий;

— пробуждение системных потоков **swapper**, **pagedaemon**, то есть регистрация отложенного вызова процедуры **wakeup**, которая перемещает дескрипторы процессов из списка «спящих» в очередь готовых потоков;

— декремент счетчика времени до отправки одного из следующих сигналов:

1) **SIGALRM** — сигнал, посылаемый потоку по истечении времени, которое задано с помощью функции **alarm()**;
2) **SIGPROF** — сигнал, посылаемый потоку по истечении времени, которое задано в таймере;
3) **SIGVTALRM** — сигнал, посылаемый потоку по истечении времени, которое задано в таймере.

Обработчик прерывания от системного таймера **по кванту** посылает сигнал **SIGXCPU** текущему потоку при истечении кванта.

1.2 Windows

Обработчик прерывания от системного таймера выполняет несколько функций.

По **тику**:

— инкремент счетчика тиков;

— декремент кванта;

— декремент счетчиков отложенных задач.

По **главному тику**:

— постановка в очередь DCP (Deferred Procedure Call) отложенного вызова обработчика ловушки профилирования ядра;

— 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.

Обработчик прерывания от системного таймера **по кванту** инициализирует диспетчеризацию потоков — добавление соответствующего объекта в очередь **DPS**.

2 Пересчет динамических приоритетов

В ОС семейств UNIX/Linux и Windows динамически пересчитываться могут только приоритеты пользовательских процессов.

2.1 UNIX/Linux

Приоритет процесса — это целое числом, находящееся в диапазоне от 0 до 127. Чем меньше число, тем выше приоритет процесса. Приоритеты ядра находятся в диапазоне от 0 до 49, они зарезервированы. Приоритеты прикладных задач в диапазоне от 50 до 127.

Дескриптор процесса **struct proc** содержит следующие поля, которые относятся к приоритету процесса:

— **p_runpri** — приоритет выполнения процесса в текущий момент времени;

— **p_slppri** — приоритет состояния ожидания;

— **p_usrpri** — приоритет режима задачи;

— **p_cpu** — 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. В таблице 2.1 приведены значения приоритетов сна для некоторых событий в системе 4.3BSD.

Можно выделить следующие особые ситуации, связанные с изменением полей p_usrpri, 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.

Таблица 2.1 – 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

Изменение приоритета в режиме задачи зависит от двух факторов:

— фактора «любезности» (**p_nice**);

— последней измеренной величины использования процессора (**p_estcpu**).

Фактор «любезности» — целое число в диапазоне от 0 до 39 (по умолчанию 20). Чем меньше значение фактора «любезности», тем выше приоритет процесса.

Каждую секунду ядро системы инициализирует отложенный вызов процедуры **schedcpu()**, которая уменьшает значение **p_runpri** каждого процесса исходя из фактора «полураспада» (decay factor). В системе 4.3BSD

Таблица 2.2 – Таблица приоритетов 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
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

фактор «полураспада» рассчитывается по формуле 2.1:

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

где $load\_average$ — 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.

Если процесс в последний раз использовал большое количество процессорного времени, то его **p_estcpu** будет увеличен. Это приведет к росту значения **p_usrpri**, из чего последует понижение приоритета. Чем дольше процесс простаивает в очереди на выполнение, тем больше фактор «полураспада» уменьшает его **p_estcpu**, 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.

2.2 Windows

В Windows процессу при создании назначается базовый приоритет. Процесс по умолчанию наследует свой базовый приоритет у того процесса, который его создал. Относительно базового приоритета процесса потоку назначается относительный приоритет.

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

В Windows используется 32 уровня приоритета, которые описываются целыми числами от 0 до 31:

— 0 — зарезервирован для процесса обнуления страниц;

— от 0 до 15 — динамически изменяющиеся уровни;

— от 16 до 31 — уровни реального времени;

— 31 — наивысший приоритет.

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

Сначала Windows API систематизирует процессы по классам приоритетов:

— 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 в рамках процессов:

— 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)

В таблице 2.3 показано соответствие между приоритетами Windows API и ядра системы.

Таблица 2.3 – 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 повышает текущий приоритет потока в динамическом диапазоне (от 1 до 15) вследствие следующих причин:

— повышение приоритета владельцем блокировки;

— повышение приоритета после завершения операции ввода/вывода;

— повышение приоритета вследствие ввода из пользовательского интерфейса;

— повышение приоритета вследствие длительного ожидания ресурса исполняющей системы;

— повышение вследствие ожидания объекта ядра (семафора, мьютекса, объекта «событие»);

— повышение приоритета в случае, когда поток, готовый к выполнению, не был запущен в течение длительного времени;

— повышение приоритета проигрывания мультимедиа службой планировщика **MMCSS** (таблица 2.4).

Рекомендуемое повышение значения приоритета кода режима ядра, вызывающего такие функции, как KeReleaseMutex, KeSetEvent и KeReleaseSemaphore, является 1.

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

Таблица 2.5 – 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

ВЫВОД

Функции обработчика прерывания от системного таймера в ОС семейств UNIX/Linux и Windows схожи, так как системы обоих семейств являются системами разделения времени с вытеснением и динамически пересчитываемыми приоритетами. Общие основные функции обработчика прерывания от системного таймера:

— декремент кванта потока в UNIX/Linux и процесса в Windows;

— инициализация отложенных действий, таких как запуск планировщика;

— декремент счетчиков времени.