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 Завойских Е.В.

Группа ИУ7-53Б

Преподаватель Рязанова Н. Ю.

Москва — 2023 г.

1 Функции обработчика прерывания от системного таймера

1.1 Операционные системы семейства Unix

По тику:

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

По главному тику:

- инициализация отложенного вызова функции планировщика;
- инициализация отложенного вызова процедуры wakeup, которая меняет состояние процесса со «спящего» на «готовый к выполнению»;
- пробуждение системных процессов, таких как swapper;
- декремент счетчика времени, которое осталось до отправления одного из следующих сигналов:
  - SIGALRM – сигнал, посылаемый процессу по истечении времени, 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.

По кванту:

- отправка сигнала SIGXCPU текущему процессу, если он превысил выделенную для него квоту процессорного времени.

1.2 Операционные системы семейства Windows

По тику:

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

По главному тику:

- инициализация диспетчера настройки баланса путем освобождения объекта «событие», на котором он ожидает.

По кванту:

- инициализация диспетчеризации 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

Очередь процессов, готовых к выполнению, формируется согласно приоритетам процессов и принципу вытесняющего циклического планирования, то есть сначала выполняются процессы, имеющие больший приоритет, а процессы с одинаковыми приоритетами выполняются циклически друг за другом в течение кванта времени. 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 от 0 до 127. Чем меньше число, тем выше приоритет процесса. Приоритеты от 0 до 49 зарезервированы ядром операционной системы, пользовательские процессы могут обладать приоритетом от 50 до 127.

Приоритеты пользовательских процессов могут изменяться во времени в зависимости от следующих факторов:

- результат последнего измерения степени загруженности CPU процессом;
- фактор любезности: чем меньше его значение, тем выше приоритет процесса (только суперпользователь может поменять значение фактора с помощью системного вызова nice).

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 **p_pri** может быть повышено планировщиком для выполнения процесса в режиме ядра. При этом **p_usrpri** будет использоваться для хранения приоритета, который будет назначен процессу при возврате в режим задачи.

Ядро системы связывает приоритет сна с событием или ожидаемым ресурсом, из-за которого процесс может быть заблокирован. Когда процесс «просыпается», ядро устанавливает в поле **p_pri** значение приоритета сна, зависящее от события или ресурса, по которому произошла блокировка.

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

Системы разделения времени пытаются выделить процессорное время так, чтобы конкурирующие процессы получили его примерно в равных количествах. Такой подход требует слежения за использованием процессора каждым из процессов. При инициализации процесса поле **p_cpu** равно нулю. На каждом тике обработчик прерывания увеличивает **p_cpu** текущего процесса на единицу, вплоть до максимального значения — 127. Каждую секун-

ду обработчик прерывания инициализирует отложенный вызов процедуры schedcpu(), которая уменьшает значение **p_pri** каждого процесса исходя из фактора «полураспада», который вычисляется 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.

Кроме того, 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**, из чего следует понижение 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 приоритет. Потоку назначается приоритет относительно приоритета процесса. 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 есть диспетчер настройки баланса, который раз в секунду сканирует очередь готовых потоков. Если обнаружены потоки, ожидающие выполнения более 4 секунд, диспетчер 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 позиций 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 двух 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).

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

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:

- 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

- повышение вследствие событий планировщика или диспетчера;
- повышение приоритета владельца блокировки;
- 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

Потоки, на которых выполняются различные мультимедийные приложения должны выполняться с минимальными задержками. В Windows драйвер MMCSS (*MultiMedia Class Scheduler Service*) временно повышает приоритет таких потоков.

Одно из наиболее важных свойств для планирования потоков — категория планирования, определяющия приоритет потоков, зарегестрированных в MMCSS (таблица 2.4). Функции MMCSS временно повышают приоритет потоков до уровня, соответствующего их категориям планирования. Затем их приоритет снижается до уровня, соответствующего категории Exhausted для того, чтобы другие потоки могли получить ресурс.

Таблица 2.4 – Категории планирования

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

2.2.2 IRQL

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.

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. Тот, в свою очередь, поднимает IRQL и вызывает процедуру обработки прерывания. После выполнения этой процедуры диспетчер прерываний понижает IRQL-уровень процессора до значения, на котором он был до возникновения прерывания, а затем загружает сохраненное состояние машины. Прерванный поток продолжает выполнение с того места, в котором оно было прервано.

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

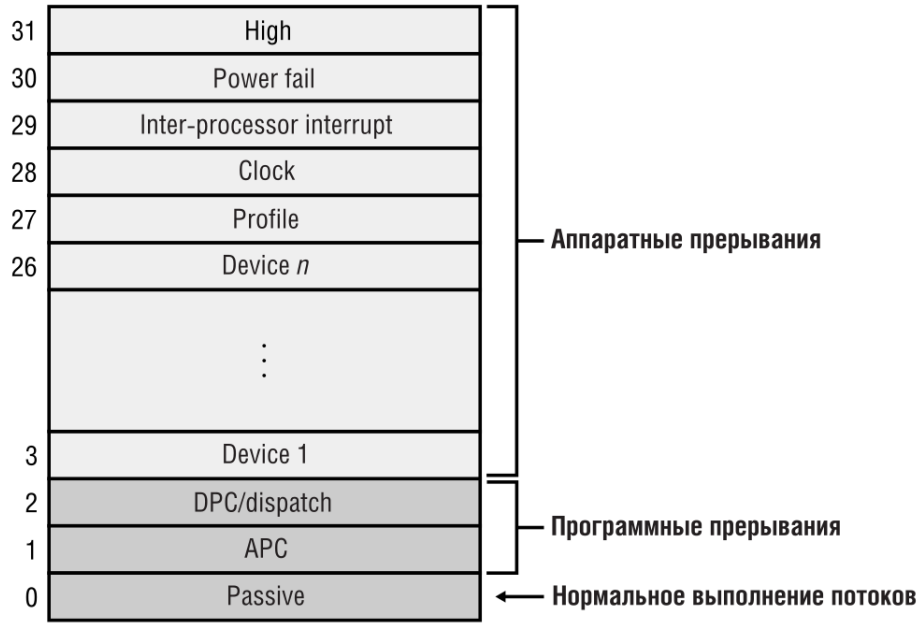

Рисунок 2.1 – Уровни запросов прерываний

Вывод

ОС семейств Unix и Windows — системы разделения времени с динамическими приоритетами и вытеснением, поэтому обработчик прерывания от системного таймера в этих системах выполняет схожие функции:

- инкремент счетчиков тиков;
- декремент кванта;
- инициализация отложенных действий, которые относятся к работе планировщика, например, пересчет приоритетов.

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