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 дисциплине "Операционны системы"

Тема Обработчик прерываний от системного таймера в Windows и Unix

Студент Светличная А.А.

Группа ИУ7-53Б

Преподаватель Рязанова Н.Ю.

Москва — 2022 г.

1 Функции обработчика прерывания от системного таймера

Обработчик прерывания от системного таймера в системе имеет наивысший приоритет, по этой причине не может быть прерван ни одним другим процессом. Значит, обработчик преывания от системного таймера должен завершаться как можно быстрее.

1.1 Unix

По тику:

- инкремент счетчика тиков аппаратного таймера;
- обновление статистики использования процессора текущим процессом (инкремент поля p_cpu структуры proc);
- декремент кванта текущего потока;
- декремент счетчика времени до отправления на выполнение отложенных вызовов (при достижении счетчиком нуля происходит выставление флага для обработчика отложенного вызова).

По главному тику:

- пробуждение системные процессы, («пробуждает» тут понимается так: регистрация отложенного вызова процедуры wakeup, которая перемещает дескрипторы процессов из списка «спящие» в очередь «готовы к выполнению»);
- регистрирация отложенных вызовы функции, которые относятся к работе планировщика (в системе SVR4 можно зарегестрировать отложенный вызов с помощью timeout(void (*fn)(), caddr_t arg, long delta); где fn() – функция, которую необходимо запустить, arg – аргументы, которые получит fn(), delta – временный интервал (в тиках процессора) через который fn() должна быть вызван);

- декремент счетчик времени, которое осталось до отправления одного из следующих сигналов:
  - SIGVTALRM – сигнал, посылаемый процессу по истечении времени, заданного в “виртуальном” таймере;
  - SIGPROF – сигнал, посылаемый процессу по истечении времени заданного в таймере профилирования;
  - SIGALRM – сигнал, посылаемый процессу по истечении времени, предварительно заданного функцией alarm().

По кванту:

- отправка текущему процессу сигнала SIGXCPU, если он израсходовал выделенный ему квант времени (при получении сигнала обработчик сигнала прерывает выполнение процесса).

1.2 Windows

По тику:

- инкремент счетчика системного времени;
- декремент счетчиков времени отложенных задач;
- декремент кванта текущего потока (декремент происходит на величину, равную количествву тактов процессора, произошедших за тик);
- Если активен механизм профилирования ядра, то инициализация отложенного вызова обработчика ловушки профилирования ядра с помощью постановки объекта в очередь DPC (обработчик ловушки профилирования регистрирует адрес команды, выполнявшейся на момент прерывания).

По главному тику:

- освобождение объекта «событие», которое ожидает диспетчер настройки баланса (диспетчер настройки баланса по событию от таймера сканирует очередь готовых процессов и повышает приоритет процессов, которые находились в состоянии ожидания дольше 4 секунд).

По кванту:

- инициация диспетчеризации потоков (добавление соответствующего объекта в очередь DPC).

2 Пересчет динамических приоритетов

В ОС семейства UNIX и в ОС семейства Windows только приоритеты пользовательских процессов могут динамически пересчитываться.

2.1 Unix

Очередь процессов, готовых к выполнению, формируется согласно приоритетам и принципу вытесняющего циклического планирования, то есть сначала выполняются процессы с большим приоритетом, а процессы с одинаковым приоритетом выполняются в течении кванта времени друг за другом циклически. В случае, если процесс с более высоким приоритетом поступает в очередь процессов, готовых к выполнению, планировщик вытесняет текущий процесс и предоставляет ресурс более приоритетному процессу. Приоритет процесса задается любым целым числом, которое лежит 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 который устанавливается пользователем.

Поля p_pri и p_usrpri применяются для различных целей. 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. Процессу, ожидающему недоступного в данный момент ресурса, система определяет значение приоритета сна, выбираемое ядром из диапазона системных приоритетов и связанное с событием, вызвавшим это состояние. В таблице 2.1 приведены значения приоритетов сна для систем 4.3BSD UNIX и SCO UNIX (OpenServer 5.0). Направление роста значений приоритета для этих систем различно — в BSD UNIX большему значению соответствует более низкий приоритет. Когда процесс завершил выполнение системного вызова и находится в состоянии возврата в режим задачи, его приоритет сбрасывается обратно в зачение текущего приоритета в режиме задачи. Измененный таким образом приоритет может оказаться ниже, чем приоритет какого-либо иного запущенного процесса, в этом случае ядро системы произведет переключение контекста.

Таблица 2.1 – Приоритеты сна в ОС 4.3BSD

Событие	Приоритет 4.3BSD UNIX	Приоритет SCO UNIX
Ожидание загрузки в память сегмента/страницы	0	95
Ожидание индексного дексриптора	10	88
Ожидание ввода-вывода	20	81
Ожидание буфера	30	80
Ожидание терминального ввода		75
Ожидание терминального вывода		74
Ожидание завершения выполнения		73
Ожидание события — низкоприоритетное состояние сна	40	66

Приоритеты в режиме задачи зависят от следующих факторов.

- "Любезности— это целое число в диапазоне от 0 до 39 со значением 20 по умолчанию. Увеличение значения приводит к уменьшению приоритета. Пользователи могут повлиять на приоритет процесса при

помощи изменения значений этого фактора, но только суперпользователь может увеличить приоритет процесса. Фоновые процессы автоматически имеют более высокие значения этого фактора.

- Последней измеренной величины использования процессора.

Системы разделения времени пытаются выделить процессорное время таким образом, чтобы конкурирующие процессы получили его примерно в равных количествах. Такой подход требует слежения за использованием процессора каждым из процессов. Каждую секунду ядро системы инициализирует отложенный вызов процедуры *schedcpu*(), которая уменьшает значение p_pri каждого процесса исходя из фактора "полураспада"(в системе 4.3BSD считается 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.

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

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

Таким образом, если процесс в последний раз использовал большое количество процессорного времени, то его p_cpu будет увеличен. 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, что приводит к повышению его приоритета. Такая схема предотвращает бесконечное откладывание низкоприоритетных 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. Относительно базового приоритета процесса потоку назначается относительный приоритет. Планирование осуществляется только на основании приоритетов потоков, готовых к выполнению: если поток с более высоким приоритетом становится готовым к выполнению, поток с более низким приоритетом вытесняется планировщиком. По истечению кванта времени текущего потока, ресурс передается самому приоритетному потоку в очереди готовых к выполнению.

В Windows используется 32 уровня приоритета, от 0 до 31:

- от 0 до 15 — 16 изменяющихся уровней, из которых уровень 0 зарезервирован для потока обнуления страниц;
- от 16 до 31 — 16 уровней реального времени.

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).

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 в таблице 2.2.

Таблица 2.2 – 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

В Windows также включен диспетчер настройки баланса, который активизируется каждую секунду для возможной инициации событий, связанных с планированием и управлением памятью. 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 готовых потоков.

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

Планировщик может повысить текущий приоритет потока в динамическом диапазоне (от 1 до 15) вследствие следующих причин:

- повышение вследствие событий планировщика или диспетчера (сокращение задержек);
- повышение вследствие завершения ввода-вывода (подходящее значение для повышения зависит от драйвера устройства таблица 2.3);
- повышение, связанные с завершением ожидания;
- повышение приоритета владельца блокировки;
- повышение при ожидании ресурсов исполняющей системы;
- повышение приоритета потоков первого плана после ожидания;
- повышение приоритета после пробуждения GUI-потока;
- повышения приоритета, связанные с перезагруженностью центрального процессора.

Таблица 2.3 – Рекомендуемые значения повышения приоритета

Устройство	Приращение
Жесткий диск, привод компакт дисков, параллельный порт, видеоустройство	1
Сеть, почтовый ящик, именованный канал, последовательный порт	2
Клавиатура, мышь	6
Звуковое устройство	8

Категории планирования представлены в таблице 2.4. Функции MMCSS временно повышают приоритет потоков, зарегистрированных с MMCSS до уровня, который соответствует категории планирования. Потом их приоритет снижается до уровня, соответствующего категории планирования Exhausted, для того, чтобы другие потоки тоже 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

3 Заключение

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

- декремент счетчиков времени;
- декремент кванта;
- инициализация отложенных действий, которые относятся к работе планировщика.

ОС семейства UNIX и Windows — это системы разделения времени с вытеснением и динамическими приоритетами.

В ОС семейства UNIX приоритет пользовательского процесса (процесса в режиме задач) может динамически пересчитываться, в зависимости от фактора «любезности», p_cpu (результат последнего измерения использования процессора) и базового приоритета (PUSER). Приоритеты ядра — фиксированные величины.

В ОС семейства Windows при создании процесса ему назначается базовый приоритет, относительно базового приоритета процесса потоку назначается относительный приоритет, таким образом, у потока нет своего приоритета. Приоритет потока пользовательского процесса может быть динамически пересчитан.