1. Функции обработчика прерывания от системного таймера

1.1. Windows

По тику:

— инкремент счётчика реального времени;

— декремент кванта на величину, равную количеству тактов процессора, произошедших за тик;

— декремент счётчиков времени до выполнения отложенных задач.

По главному тику:

— инициация диспетчера настройки баланса путём сбрасывания объекта «событие», на котором он ожидает.

По кванту:

— инициализация диспетчеризации потоков путём постановки соответствующего объекта KiDispatchInterrupt в очередь DPC.

1.2. Unix/Linux

По тику:

— инкремент счётчика времени с момента запуска системы (SVR4, переменная lbolt);

— декремент кванта активного процесса;

— декремент счетчика времени до отправки на выполнение отложенного вызова (при достижении счетчиком нулевого значения выставляется флаг для обработчика отложенных вызовов).

По главному тику:

— инициализация отложенных действий, относящихся к работе планировщика;

— пробуждение системных процессов, таких как swapper и pagedaemon;

— декремент счетчика времени, оставшегося до посылки одного из сигна- лов:

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:

— посылка сигнала SIGXCPU активному процессу, если он израсходовал выделенный ему квант процессорного времени.

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, другие имеют статические приоритеты.

2.1. Windows

В ОС семейства Windows процессу при создании назначается базовый приоритет. Относительно базового приоритета процесса потоку назначается относительный приоритет. Планирование осуществляется на основе приоритетов потоков, готовых к выполнению. Поток с более высоким приоритетом вытесняет поток с более низким приоритетом. По истечении кванта времени текущего потока, ресурс передается самому приоритетному потоку в очереди готовых к выполнению.

В ОС семейства Windows определено 32 уровня приоритетов:

— от 0 до 15 — изменяющиеся уровни (уровень 0 зарезервирован для потока обнуления страниц);

— от 16 до 31 — уровни реального времени.

Система не повышает приоритет потоков с базовым уровнем приоритета от 16 до 31. Только потоки с базовым приоритетом от 0 до 15 получают динамический приоритет. Потоки реального времени имеют статический приоритет

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);

— простоя (idle, 1).

Затем назначается относительный приоритет 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);

— простоя (idle, -15).

В таблице 1 приведено соответствие между приоритетами Windows API и ядра Windows.

Таблица 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

Диспетчер настройки баланса 1 раз в секунду сканирует очередь готовых потоков, если обнаружены потоки, ожидающие выполнения более 4 секунд, он 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.

Приоритет потока в может быть изменён вследствие следующих причин:

— повышение после завершения операции ввода-вывода;

— повышение вследствие ввода из пользовательского интерфейса;

— повышение вследствие длительного ожидания ресурса исполняющей системы;

— повышение в случае, когда готовый к запуску поток не был запущен в течение определенного времени;

— повышения приоритета, связанные с завершением ожидания объекта ядра на 1 (семафор, мьютекс);

— повышение приоритета после пробуждения GUI-потока;

— повышение приоритета проигрывания мультимедиа службой планировщика MMCSS.

В таблице 2 приведены рекомендуемые значения повышения приоритета потока для устройств ввода-вывода. Эти значения являются рекомендуемыми, так как именно драйвер устройства указывает повышение приоритета при вызове функции ядра.

Таблица 2. Рекомендуемые значения повышения приоритета.

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

2.2. MMCSS

Для обеспечения минимальных задержек при работе мультимедийных приложений в Windows используется служба MMCSS. Приложения, воспроизводящие мультимедиа, регистрируются с одной из задач, с которой работает MMCSS:

— аудио;

— захват;

— распределение;

— игры;

— проигрывание;

— аудио профессионального качества;

— задачи администратора многоэкранного режима.

Одно из наиболее важных свойств для планирования потоков называется категорией планирования (Scheduling Category), которое является первичным фактором, определяющим приоритет потоков, зарегистрированных с MMCSS. Различные категории планирования представлены в таблице 3.

Таблица 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

Механизм, положенный в основу MMCSS, повышает приоритет потоков внутри зарегистрированного процесса до уровня, соответствующего их категории планирования и относительного приоритета внутри этой категории на гарантированный срок. Затем он снижает категорию этих потоков до Exhausted, чтобы другие, не относящиеся к мультимедийным приложениям потоки, также получили шанс на выполнение.

2.3. IRQL

Контроллеры прерываний устанавливают приоритетность прерываний, но 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 (рисунок 1), где более высоким номерам соответствуют прерывания с более высоким приоритетом.

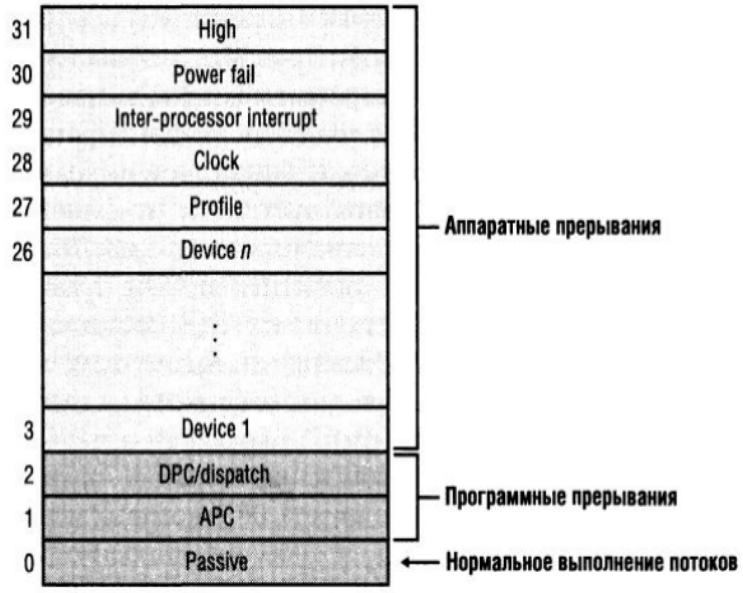

Рис. 1. Уровни запросов прерываний IRQL для архитектуры x86

Прерывания обслуживаются в порядке их приоритета, и прерывания с более высоким уровнем приоритета получают преимущество в обслуживании. При возникновении прерывания с высоким уровнем процессор сохраняет состояние прерванного потока и запускает связанный с прерыванием диспетчер системных прерываний.

2.4. Unix/Linux

Согласно приоритетам процессов и принципу вытесняющего циклического планирования формируется очередь готовых к выполнению процессов. В

первую очередь выполняются процессы с большим приоритетом. Процессы с одинаковыми приоритетами выполняются в течение кванта времени, циклически друг за другом. Если процесс, имеющий более высокий приоритет, поступает в очередь процессов, готовых к выполнению, то он вытеснит текущий процесс.

Приоритет задается целым числом, лежащим в диапазоне от 0 до 127 (чем меньше число, тем выше приоритет). Приоритеты ядра находятся в диапазоне от 0 до 49, а приоритеты прикладных задач в диапазоне от 50 до 127. Приоритеты ядра являются статическими величинами, а приоритеты прикладных задач могут изменяться во времени в зависимости от двух факто- ров: фактора «любезности» и последней измеренной величиной использования процессора.

Фактор «любезности» — это целое число в диапазоне от 0 до 39 (чем меньше значение фактора «любезности», тем выше приоритет процесса). Фактор«любезности» процесса может быть изменен только суперпользователем.

Дескриптор процесса proc содержит следующие поля, относящиеся к приоритету процесса:

— p_pri — текущий приоритет планирования;

— p_usrpri — приоритет процесса в режиме задачи;

— p_cpu — результат последнего измерения использования процессора;

— p_nice — фактор «любезности», устанавливаемый пользователем.

У процесса, находящегося в режиме задачи, значения p_pri и p_usrpri равны. Когда процесс просыпается, его приоритет временно повышается согласно таблице приоритетов сна (табл. 4). Планировщик использует p_usrpri для хранения приоритета, который будет назначен процессу при возвращении в режим задачи, а p_pri — для хранения временного приоритета для выполнения в режиме ядра.

При создании процесса поле p_cpu инициализируется нулем. На каждом тике обработчик таймера увеличивает значение p_cpu на единицу, до максимального значения, равного 127. Каждую секунду обработчик прерывания инициализирует отложенный вызов процедуры schedcpu(), которая уменьшает значение p_cpu каждого процесса исходя из фактора полураспада. В системе 4.3BSD для расчёта фактора полураспада используется формула:

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

где load_average — среднее количество процессов, находящихся в состоянии готовности к выполнению, за последнюю секунду.

Таблица 4. Приоритеты сна в системах 4.3BSD UNIX и SCO UNIX

Событие	Приоритет 4.3BSD UNIX	Приоритет SCO UNIX
Ожидание загрузки страницы/сегмента	0	95
Ожидание индексного дескриптора	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	40	66

Процедура schedcpu() также пересчитывает приоритеты режима задачи всех процессов по формуле, где PUSER — базовый приоритет в режиме задачи, равный 50:

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

Таким образом, если процесс до вытеснения другим процессом использовал большое количество процессорного времени, его p_cpu будет увеличен, что приведет к увеличению значения p_usrpri и, следовательно, к понижению 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 позволяет исключить бесконечное откладывание низкоприоритетных процессов.

2.5. Потоки в Linux

Современное ядро Linux является вытесняемым. При создании процесса в ОС Linux он рассматривается как главный поток. Планировщик может работать с потоками реального времени, которые имеют приоритеты в диапазоне от 0 до 99, пользовательские потоки и потоки в режиме ядра имеют приоритеты в диапазоне от 100 до 139. Поток реального времени может вытеснить поток, находящийся в режиме ядра. Потоки реального времени имеют статические приоритеты.

Для потоков реального времени в системе существуют следующие классы планирования:

— SCHED_RR — потоку выделяется квант времени, он может быть вытеснен потоком с более высоким приоритетом;

— SCHED_FIFO — поток выполняется, пока не освободит процессор или не будет вытеснен потоком с более высоким приоритетом;

— SCHED_DEADLINE — поток должен закончить работу к определённому моменту времени.

Для потоков пользовательского режима в системе существуют следующие классы планирования:

— SCHED_OTHER — приоритет потока определяется аналогично старому ядру Linux;

— SCHED_BATCH — вычислительные потоки;

— SCHED_IDLE — фоновые потоки.

Структура task_struct содержит поля, относящиеся к приоритетам: static_prio, rt_priority — приоритеты, основанные на классе планирования, normal_prio, prio, используемый планировщиком для принятия решения о том, какой из потоков отправить на выполнение.

Значение static_prio рассчитывается следующим образом:

$$static\_prio = 120 + nice, \quad (1)$$

где nice — фактор любезности потока в диапазоне от -20 до 19.

Диапазон приоритетов потоков реального времени отображается в приоритеты ядра по формуле:

$$MAX\_RT\_PRIO - 1 - rt\_priority, \quad (2)$$

где MAX_RT_PRIO равно 100.

Значение normal_prio для потоков реального времени рассчитывается аналогично формуле (2), для динамических потоков значение равное static_prio. Когда поток просыпается после блокирования в системном вызове, его приоритет временно повышается, планировщик использует normal_prio для хранения приоритета, который будет назначен потоку при возвращении в режим задачи.

Выводы

Операционные системы семейств Windows и UNIX являются системами разделения времени с динамическими приоритетами и вытеснением, поэтому обработчик прерывания от системного таймера выполняет в этих системах схожие функции:

— инициализация отложенных действий, которые относятся к работе планировщика;

— декремент кванта;

— инкремент счётчиков времени, декремент «будильников».

Пересчёт динамических приоритетов осуществляется только для пользовательских потоков для того, чтобы избежать их бесконечного откладывания. В ОС семейства UNIX при пересчёте приоритетов учитывается время простоя потока в очереди готовых потоков и полученное процессорное время. В ОС семейства Windows пересчёт приоритетов осуществляется путём сканирования очереди готовых потоков.