|  |  |
| --- | --- |
|  | **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 Информатики и систем управления

КАФЕДРА Программное обеспечение ЭВМ и информационыне технологии (ИУ7)

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1(часть №2)**

Тема: Прерывание таймера в OC Windows и Unix

Студент: ­ Фам М. Х.

Группа: ИУ7-52Б

Преподаватель: Рязанова Н. Ю.

*Москва, 2023г.*

# Функции обработчика прерывания системного таймера в системах разделения времени

## UNIX/Linux

**По тику**

1. Инкремент счетчика времени с момента запуска системы (SVR4, переменная lbolt)
2. Обновляет статистику использования процессора текущим процессом.
3. Декремент кванта текущего процесса.
4. Декремент счетчиков времени и при достижении счетчиками нулевого значение установка флага для обработчиков отложенных вызовов;
5. Инкремент счетчика процессорного времени, полученного процессором в режиме задачи и в режиме ядра.

**По главному тику**

1. Пробуждение системных процессов swapper и pagedaemon.
2. Инициализация отложенных действий, относящихся к работе планировщика.
3. Декремент счетчика времени, оставшегося до отправления одного из 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**

1. Посылка текущему процессу сигнала SIGXCPU, если тот израсходовал выделенный ему квант процессорного времени

## Windows

**По тику**

* Инкремент счётчика реального времени
* Декремент кванта текущего потока на величину, равную количеству тактов процессора, произошедших за тик
* Декремент счётчиков времени отложенных задач

**По главному тику**

* Инициализация диспетчера настройки баланса путём освобождения объекта «событие»

**По кванту**

* Инициализация диспетчеризации потоков – постановка соответствующего объекта в очередь DPC

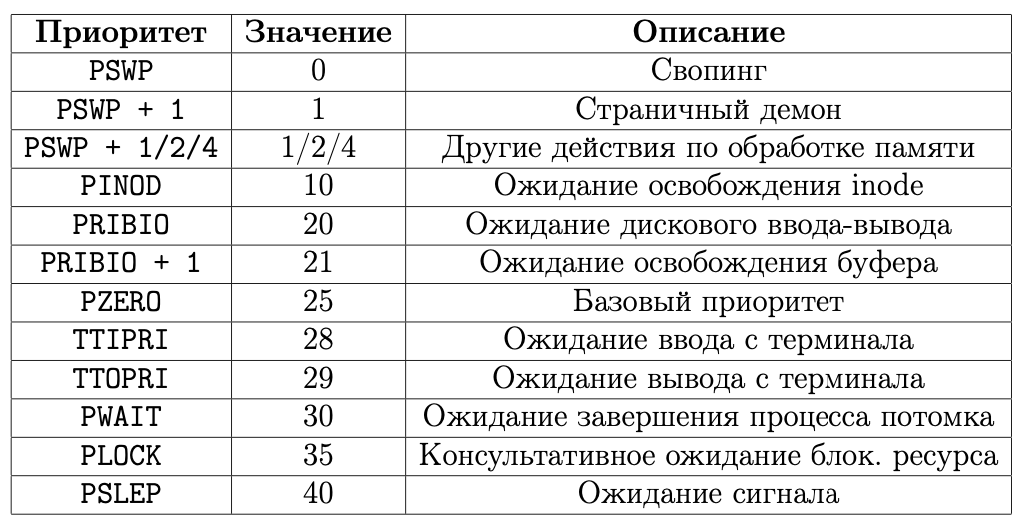
# Динамические приоритеты

Системы семейств Unix и Windows являются системами разделения времени с динамическими приоритетами и вытеснением. Динамические приоритеты могут иметь только пользовательские процессы, другие процессы имеют статические приоритеты.

## UNIX/Linux

Приоритет процесса представляет собой число от 0 до 127. Чем меньше значение, тем выше приоритет. В диапазоне от 0 до 49 находятся приоритеты ядра. В диапазоне от 50 до 127 находятся приоритеты прикладных задач.

Приоритет процесса определяется двумя факторами: «любезностью» процесса (число в диапазоне от 0 до 39 со значением 20 по умолчанию, увеличение значения приводит к увеличению значения приоритета процесса) и последней измеренной величиной использования процессора.

****Таблица 2.1 – Приоритеты сна в ОС **4.3BSD**

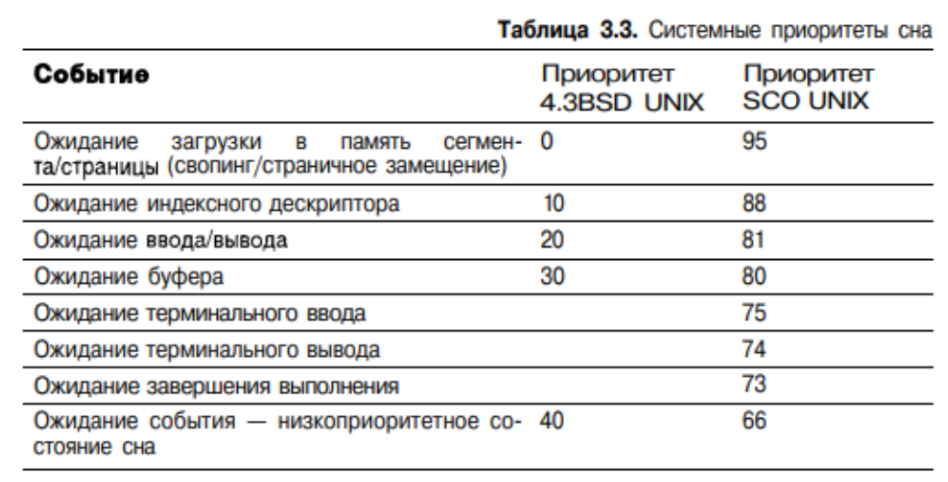
****

Рисунок 2.1 – Системные приоритеты сна

Очередь готовых к выполнению процессов формируется исходя из приоритетов процессов и принципу вытесняющего циклического планирования. В первую очередь выполняются процессы с самым высоким приоритетом, а процессы с одинаковыми приоритетами выполняются в течении кванта времени циклически друг за другом. Если процесс, имеющий более высокий приоритет, поступает в очередь процессов, готовых к выполнению, то планировщик вытесняет текущий процесс и предоставляет ресурс более приоритетному.

Традиционное ядро UNIX является строго невытесняемым. Если процесс выполняется в режиме ядра, то ядро не заставит такой процесс уступить процессор какому-либо процессу с высоким приоритетом. Это сделано для того, чтобы поддерживать процессы реального времени.

В современных системах UNIX/Linux ядро является вытесняющим, т.е. процесс в режиме ядра может быть вытеснен более приоритетным процессом.

Структура proc содержит следующие поля, относящиеся к приоритетам:

1. p\_pri — текущий приоритет планирования
2. p\_usrpri — приоритет режима задачи
3. p\_cpu — результат последнего измерения использования процессора
4. p\_nice — значение «любезности» процесса

При создании процесса поле p\_cpu инициализируется нулем. На каждом тике обработчик таймера увеличивает поле p\_cpu текущего процесса на единицу, до максимального значения, равного 127. Каждую секунду, обработчик прерывания инициализирует отложенный вызов процедуры schedcpy(), которая уменьшает значение p\_cpu каждого процесса исходя из фактора «полураспада» (decay factor) и пересчитывает приоритеты для режима задачи всех процессов. В системе SVR3 используется фиксированное значение этого фактора, равное 0,5. В 4.3BSD используется следующая формула.

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

Для пересчета приоритетов в режиме задачи используется следующая формула.

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

Таким образом, если процесс в последний раз использовал большое количество процессорного времени, его p\_cpu будет увеличен. Это приведет к росту значения p\_usrpri и, следовательно, к понижению приоритета. Чем дольше процесс простаивает в очереди на исполнение, тем больше фактор полураспада уменьшает его p\_cpu, что приводит к повышению его приоритета. Такая схема предотвращает зависание процессов с низким приоритетом по вине операционной системы. Ее применение предпочтительнее процессам, осуществляющим много операций ввода-вывода, в противоположность процессам, производящим много вычислений.

## Windows

При создании процесса, ему назначается приоритет. Потоку назначается приоритет относительно базового приоритета процесса.

Раз в секунду диспетчер настройки баланса сканирует очередь готовых потоков. Если обнаружены потоки, ожидающие выполнения более 4 секунд, 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.

Чтобы минимизировать расход процессорного времени, диспетчер настройки баланса сканирует лишь 16 готовых потоков. Кроме того, диспетчер повышает приоритет не более чем у 10 потоков за один проход: обнаружив 10 потоков, приоритет которых следует повысить, он прекращает сканирование. При следующем проходе сканирование возобновляется с того места, где оно было прервано в прошлый раз. Наличие 10 потоков, приоритет которых следует повысить, говорит о необычно высокой̆ загруженности системы.

В Windows используется 32 уровня приоритета (целое число от 0 до 31, 31 наивысший). Эти значения разбиваются на категории следующим образом:

* приоритет 31 — наивысший;
* от 16 до 31 — процессы реального времени;
* от 0 до 15 — динамические уровни;
* 0 — зарезервирован для процесса обнуления страниц.

Уровни приоритета потоков назначаются исходя из двух разных позиций: одной̆ 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)
* Простоя — Idle (1)

Уровни приоритета потоков назначаются с двух позиций: Windows API и ядра операционной системы. Windows API сортирует процессы по классам приоритета, которые были назначены при из создании:

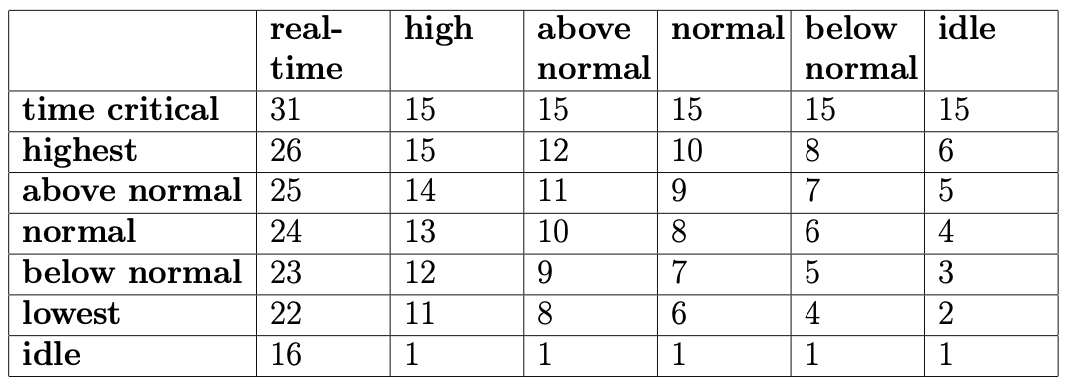
* реального времени (real-time, 4);
* высокий (high, 3);
* выше обычного (above normal, 6);
* обычный (normal, 2);
* ниже обычного (below normal, 5);
* простой (idle, 1).

Затем назначается относительный приоритет потоков в рамках 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).

Соответствие между приоритетами Windows API и ядра системы приведено в таблице 2.1.

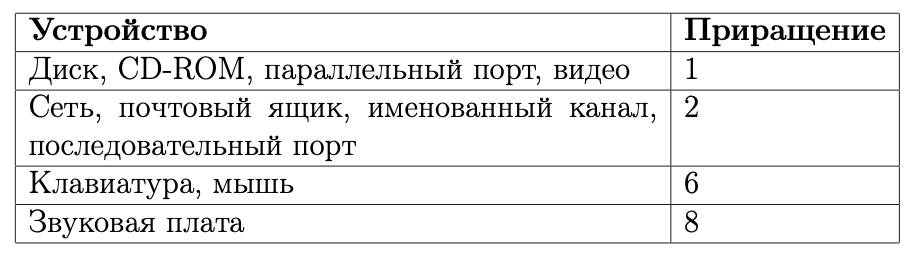
Таблица 2.2 – Соответствие между приоритетами Windows API и ядра



Текущий приоритет потока в динамическом диапазоне может быть повышен планировщиком вследствие следующих причин:

* завершение операций ввода-вывода;
* 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;

Таблица 2.3 – Рекомендуемые значения повышения приоритета



Текущий приоритет потока в динамическом диапазоне может быть понижен до базового приоритета путем вычитания всех повышений.

## MMCSS

Потоки, на которых выполняются различные мультимедийные приложения, должны выполняться с минимальными задержками. В Windows эта задача решается путем повышения приоритетов таких потоков драйвером MMCSS – MultiMedia Class Scheduler Service. Приложения, которые реализуют воспроизведение мультимедиа, указывают драйверу MMCSS задачу из списка:

1. аудио;
2. возможность использовать функции записи;
3. воспроизведение звукового или видео контента;
4. задачи администратора многооконного режима.

Одно из наиболее важных свойств для планирования потоков — категория планирования — первичный фактор определяющий приоритет потоков, зарегистрированных в MMCSS. Различные категории планирования представлены в таблице ниже.

Функции MMCSS временно повышают приоритет потоков, зарегистрированных с MMCSS до уровня, соответствующего их категориям планирования. Далее, их приоритет снижается до уровня, соответствующего категории Exhausted, для того чтобы другие потоки могли получить ресурс.

## 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 (рисунок 2.1), где более высоким номерам соответствуют прерывания с более высоким приоритетом.

Прерывания обслуживаются в порядке их приоритета. При возникновении прерывания с высоким приоритетом процессор сохраняет информацию о состоянии прерванного потока и запускает связанные с прерывание диспетчер системных прерываний.

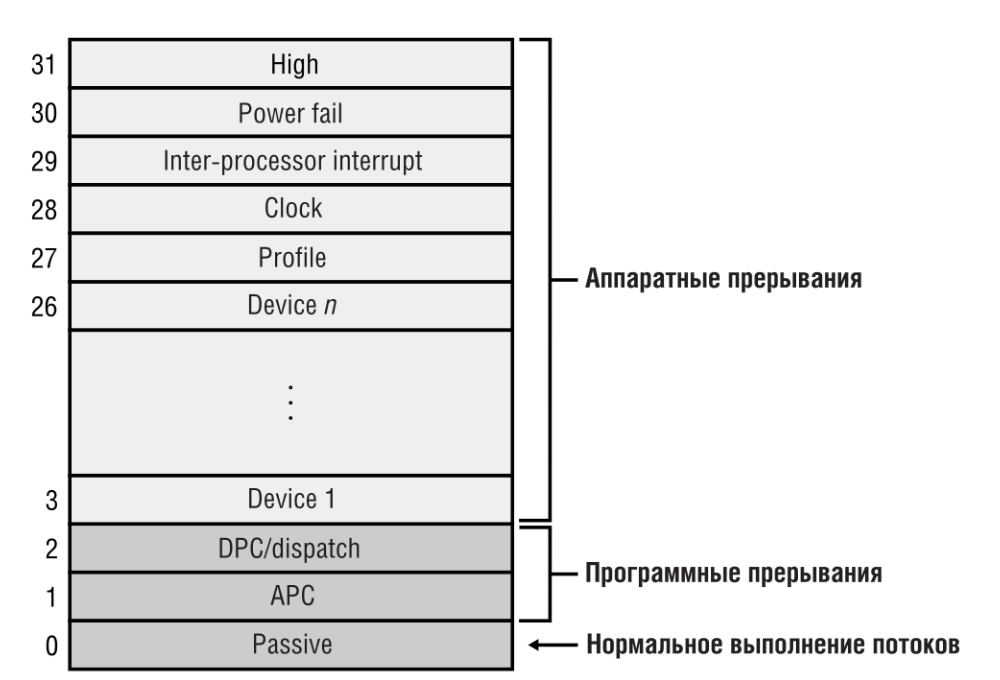


Рисунок 2.1 – Уровни запросов прерываний

# Вывод

Операционные системы семейств Unix и Windows являются системами разделения времени с динамическими приоритетами и вытеснением, поэтому функции обработчика прерывания от системного таймера в этих системах выполняют схожие задачи:

* декремент кванта в UNIX и в Windows.
* инициализация отложенных действий, которые относятся к работе планировщика (например, пересчет приоритетов).
* декремент счетчиков времени (таймеров, часов, счетчиков времени отложенных действий, будильников реального времени)

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