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»

Лабораторная работа 1

По курсу: «Операционные системы»

Функции обработчика прерывания от системного таймера

Выполнил: Мороз Д.В.,
Студент гр. ИУ7-53Б

2019 г.

Функции обработчика прерывания от системного таймера

1. В Windows

    1.1. По тику

        1.1.1. Инкремент счётчика системного времени

        1.1.2. Декремент счетчиков отложенных задач

        1.1.3. Декремент остатка кванта текущего потока

        1.1.4. Добавление процесса в очередь DPC

    1.2. По главному тику

        1.2.1. Инициализация диспетчера настройки баланса

        1.2.2. Пробуждение некоторых системных процессов

    1.3.По кванту

        1.3.1. Инициализация диспетчеризации потоков

2. В UNIX/LINUX

    2.1. По тику

        2.1.1. Ведение учета использования процессора

        2.1.2. Инкремент таймеров системы

        2.1.3. Декремент счетчика времени до отправления на выполнение отложенных вызовов

    2.2. По главному тику

        2.2.1. добавление в очередь на выполнение функций, относящихся к работе планировщика-диспетчера

        2.2.2. Декремент времени, оставшегося до отправления SIGALARM или SIGPROF или SIGVTALARM

2.3.По кванту

2.3.1. Если израсходован выделенный квант процессорного времени, посылка текущему процессу сигнала SIGXCPU

2.3.2. Пробуждение некоторых системных процессов

В Windows длительность интервала таймера зависит от аппаратной платформы и определяется HAL, а не ядром. Например, этот интервал на большинстве однопроцессорных x86 систем составляет 10 мс, а на большинстве многопроцессорных x86 систем — около 15 мс.

В UNIX продолжительность тика обычно составляет 10 мс. Во многих реализациях частота таймера хранится в специальной константе HZ, определенной в файле param.h.

Пересчет динамических приоритетов

Windows

Ядро Windows не имеет центрального потока планирования. Вместо этого, когда поток не может больше выполняться, он сам вызывает планировщик, чтобы увидеть, не освободился ли в результате его действий поток с более высоким приоритетом планирования, который готов к выполнению. Если это так, то происходит переключение потоков. Поскольку Windows является полностью вытесняющей, то есть переключение потоков может произойти в любой момент, а не только в конце кванта текущего потока.

Планирование вызывается при следующих условиях:

1. Выполняющийся поток блокируется на семафоре, мьютексе, событии, вводе/выводе и т. д.
2. Поток подает сигнал об объекте

3. Истекает квант времени потока

В Windows 32 уровня приоритета: от 0 до 31: шестнадцать уровней реального времени (от 16 до 31), шестнадцать изменяющихся уровней (от 0 до 15), из которых уровень 0 зарезервирован для потока обнуления страниц.

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 (7), Обычный — Normal (2), Ниже обычного — Below Normal (5) и 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) и Простоя — Idle (–15).

Базовый алгоритм планирования делает поиск по массиву от приоритета 31 до приоритета 0. Как только будет найден непустой список, поток выбирается сверху списка и выполняется в течение одного кванта. Если квант истекает, то поток переводится в конец очереди своего уровня приоритета и следующим выбирается верхний поток списка. Если готовых потоков нет, то процессор переходит в состояние ожидания, то есть переводится в состояние более низкого энергопотребления и ждет прерывания.

Потоки приложений обычно выполняются с приоритетами 1–15. Как правило, пользовательские приложения и службы запускаются с обычным базовым приоритетом (normal), поэтому их исходный поток чаще всего выполняется с уровнем приоритета 8.

Повышение приоритета вступает в действие немедленно и может вызвать изменения в планировании процессора. Однако если поток использует весь свой следующий квант, то он теряет один уровень приоритета и перемещается вниз на одну очередь в массиве приоритетов. Если же он использует второй полный квант, то он перемещается вниз еще на один уровень, и так до тех пор, пока не дойдет до своего базового уровня (где и останется до следующего повышения). Повышение приоритета потока в Windows применяется только для потоков с приоритетом динамического диапазона (0-15). Но каким бы ни было приращение, приоритет потока никогда не будет больше 15. Таким образом, если к потоку с приоритетом 14 применить динамическое повышение на 5 у ровней, т о его приоритет станет равным только 15 (если приоритет потока равен 15, то повысить его нельзя).

Приоритет потока повышается:

- Когда операция ввода-вывода завершается и освобождает находящийся в состоянии ожидания поток, то его приоритет повышается (чтобы он мог опять быстро запуститься и начать новую операцию ввода-вывода).
- Если поток ждал на семафоре, мьютексе или другом событии, то при его освобождении он получает повышение приоритета на два уровня, если находится в фоновом процессе, и на один уровень во всех остальных случаях.
- Если поток графического интерфейса пользователя просыпается по причине наличия ввода от пользователя, то он также получает повышение.
- Если поток, готовый к выполнению, задерживается из-за нехватки процессорного времени.

Для обеспечения поддержки мультизадачности системы, когда исполняется код режима ядра, Windows использует приоритеты прерываний IRQL. Ядро определяет стандартный набор IRQL для программных прерываний, а HAL увязывает IRQL с номерами аппаратных прерываний. Потоки обычно запускаются на уровне IRQL0 или на уровне IRQL1. Код пользовательского режима всегда запускается на пассивном уровне.

Прерывания обслуживаются в порядке их приоритета. При возникновении прерывания с высоким приоритетом процессор сохраняет информацию о состоянии прерванного потока и активизирует сопоставленный с данным прерыванием диспетчер ловушки. Последний повышает IRQL и вызывает процедуру обслуживания прерывания (ISR). После выполнения ISR диспетчер прерывания понижает IRQL процессора до исходного уровня и загружает сохраненные ранее данные о состоянии машины. Когда ядро понижает IRQL, могут начать обрабатываться ранее замаскированные прерывания с более низким приоритетом.

UNIX/LINUX

В фазе “пользователь” приоритет процесса имеет 2 составляющие: пользовательскую и системную. Значения этих составляющих задают поля дескриптора процесса p_nice и p_cpu. Начальное значение пользовательской составляющей равно константе NZERO (=20). Пользовательская составляющая может быть изменена системным вызовом nice с аргументом, определяющим величину изменения поля p_nice в пределах от 0 до NZERO для непривилегированного процесса и от –NZERO до +NZERO для привилегированного. Начальное значение системной составляющей в фазе пользователь равно 0. Ее изменение зависит от времени использования процессора. Для формирования системной составляющей приоритета используются прерывания от аппаратного таймера. Каждое прерывание по

таймеру увеличивает значение поля p_cpu на 1. Результирующий приоритет процесса в фазе «пользаватель» определяется по формуле: p_pri = (p_nice – NZERO) + (p_cpu/16) + P_USER, где P_USER - константа, по умолчанию равная 50.

Планировщик всегда выбирает процесс с наивысшим приоритетом. Приоритет процесса не является фиксированным и динамически изменяется системой в зависимости от использования вычислительных ресурсов, времени ожидания запуска и текущего состояния процесса. Если процесс готов к запуску и имеет наивысший приоритет, планировщик приостановит выполнение текущего процесса (с более низким приоритетом), даже если последний не "выработал" свой временной квант.

Приоритет процесса/потока задается любым целым числом, лежащим в диапазоне от 0 до 139, то есть существует 140 уровней приоритета(для обычных потоков и потоков реального времени). Чем меньше такое число, тем выше приоритет. В Unix приоритеты от 0 до 49 зарезервированы для ядра, следовательно, прикладные процессы могут обладать приоритетом в диапазоне 50–139.

В UNIX структура proc содержит следующие поля, относящиеся к приоритетам:

p_pri	Текущий приоритет планирования	Используется для хранения временного приоритета для выполнения в режиме ядра.
p_usrpri	Приоритет режима задачи	Используется для хранения приоритета, который будет назначен процессу при возврате в режим задачи.
p_cpu	Результат последнего измерения использования процессора	Содержит величину результата последнего сделанного измерения использования процессора процессом. Инициализируется нулем.
p_nice	Фактор nice, устанавливаемый пользователем	(в диапазоне от 0 до 39 со значением 20 по умолчанию) Увеличение значения приводит к уменьшению приоритета.