ПЛАНИРОВАНИЕ ПРОЦЕССОВ

На протяжении существования процесса выполнение его потоков может быть многократно прервано и продолжено. Переход от выполнения одного потока к другому осуществляется в результате планирования и диспетчеризации. Работа по определению того, в какой момент необходимо прервать выполнение текущего активного потока и какому потоку предоставить возможность выполняться, называется **планированием.** Планирование потоков осуществляется на основе информации, хранящейся в описателях процессов и потоков.

При планировании могут приниматься во внимание приоритет потоков, время их ожидания в очереди, накопленное время выполнения, интенсивность обращений к вводу-выводу и другие факторы.

Планирование потоков включает в себя решение двух задач:

– определение момента смены текущего активного потока;

– выбор для выполнения потока из очереди готовых потоков.

Существует множество различных алгоритмов планирования потоков, решающих приведенные задачи.

Особенности реализации планирования потоков в наибольшей степени определяют **специфику операционной системы,** в частности, является ли она системой пакетной обработки, системой разделения времени или системой реального времени.

В большинстве операционных систем универсального назначения планирование осуществляется **динамически,** то есть решения принимаются во время работы системы на основе анализа текущей ситуации.

Другой тип планирования – **статический –** может быть использован в специализированных системах, в которых весь набор одновременно выполняемых задач определен заранее, например, **в системах реального времени**. Планировщик называется статическим, если он принимает решения о планировании не во время работы системы, а заранее. Результатом работы статического планировщика является таблица, называемая расписанием, в которой указывается, какому потоку/процессу, когда и на какое время должен быть предоставлен процессор.

Диспетчеризация заключается в реализации найденного в результате планирования (динамического или статистического) решения, то есть в переключении процессора с одного потока на другой. Прежде чем прервать выполнение потока, ОС запоминает его контекст, для того, чтобы впоследствии использовать эту информацию для возобновления выполнения данного потока.

Диспетчеризация обычно сводится к:

– сохранению контекста текущего потока, который требуется сменить;

– загрузке контекста нового потока;

– запуску нового потока на выполнение.

Операция переключения контекстов ощутимо влияет на производительность вычислительной системы, поэтому модули ОС выполняют диспетчеризацию потоков совместно с аппаратными средствами процессора. В контексте потока можно выделить часть, общую для всех потоков данного процесса (ссылки на открытые файлы), и часть, относящуюся только к данному потоку (содержимое регистров, счетчик команд, режим процессора).

Планирование процессов включает в себя решение следующих задач:

- определение момента времени для смены выполняемого процесса;
- выбор процесса на выполнение из очереди готовых процессов;
- переключение контекстов «старого» и «нового» процессов.

Первые две задачи решаются программными средствами, а последняя в значительной степени аппаратно.

Все множество алгоритмов планирования можно разделить на два принципиально различных класса: **вытесняющие** и **невытесняющие**.

– *Невытесняющие (non-preemptive)* алгоритмы основаны на том, что активному потоку позволяется выполняться до тех пор, пока он сам, по собственной инициативе, не отдаст управление операционной системе для того, чтобы та выбрала из очереди другой готовый к выполнению поток.

– *Вытесняющие (preemptive)* алгоритмы – это такие способы планирования потоков, в которых решение о переключении процессора с выполнения одного потока на выполнение другого потока принимается операционной системой, а не активной задачей.

Основным различием между двумя типами алгоритмов является **степень централизации механизма планирования потоков. Вытесняющие алгоритмы** целиком сосредоточены в операционной системе. При этом операционная система сама определяет момент снятия с выполнения активного потока, запоминает его контекст, выбирает из очереди готовых потоков следующий, запускает новый поток на выполнение, загружая его контекст.

При невытесняющем мультипрограммировании механизм планирования распределен между операционной системой и прикладными программами.

Прикладная программа, получив управление от операционной системы, сама передает управление ОС с помощью какого-либо системного вызова, когда посчитает это необходимым.

Такой механизм создает проблемы, как для пользователей, так и для разработчиков приложений. Если приложение тратит слишком много времени на выполнение какой-либо работы, например на форматирование диска, пользователь не может переключиться с этой задачи на другую задачу.

Почти во всех современных операционных системах, ориентированных на высокопроизводительное выполнение приложений (UNIX, Windows NT/2000), реализованы вытесняющие алгоритмы планирования потоков (процессов).

Среди вытесняющих алгоритмов рассмотрим подробнее две группы наиболее часто встречающихся алгоритмов: **алгоритмы, основанные на квантовании**, и **алгоритмы, основанные на приоритетах.**

Алгоритмы с квантованием.

Каждому потоку предоставляется **квант** времени, в течение которого поток может выполняться на процессоре. По истечении кванта операционная система переключает процессор на следующий поток в очереди.

Алгоритмы с приоритетами.

Каждому потоку назначается **приоритет** (priority) – целое число, обозначающее степень привилегированности потока. Операционная система при наличии нескольких готовых к выполнению потоков выбирает из них поток с наибольшим приоритетом.

В соответствии с алгоритмами, **основанными на квантовании,** смена активного процесса происходит, если,

- процесс завершился и покинул систему;
- произошла ошибка;
- процесс перешел в состояние ОЖИДАНИЕ;
- исчерпан квант процессорного времени, отведенный данному процессу.

Процесс, который исчерпал свой квант, переводится в состояние ГОТОВНОСТЬ и ожидает, когда ему будет предоставлен новый квант процессорного времени, а на выполнение в соответствии с определенным правилом выбирается новый процесс из очереди готовых.

Таким образом, ни один процесс не занимает процессор надолго, поэтому квантование широко используется в **системах разделения времени.**

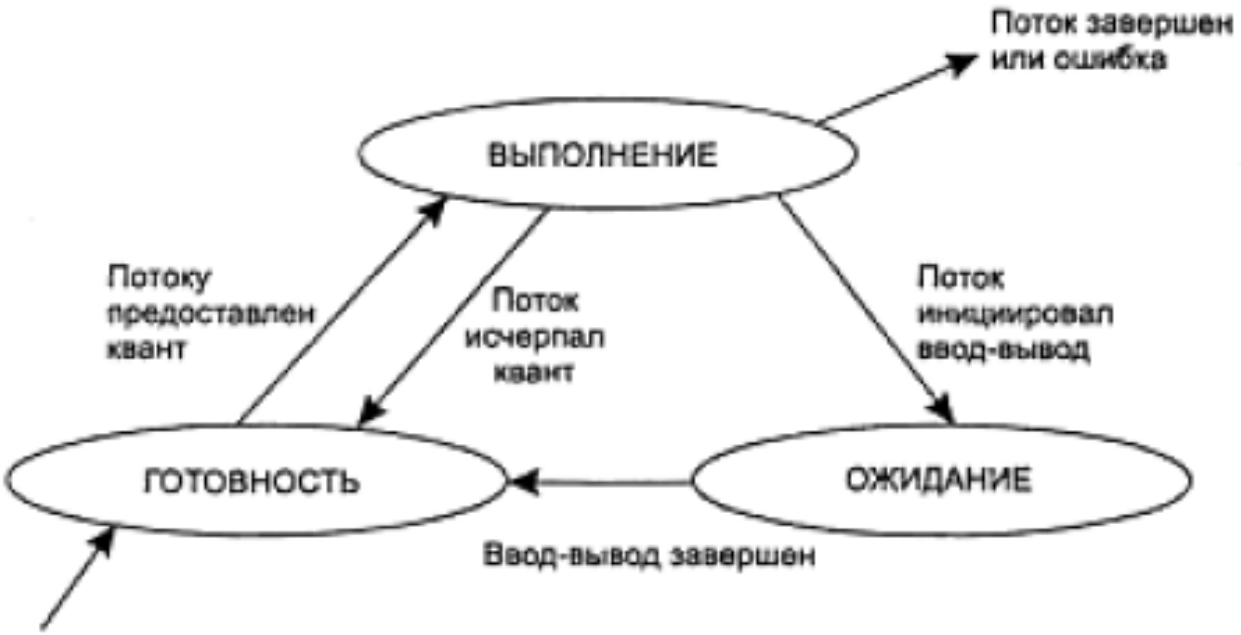


Рисунок 3.4 – Граф состояний потока в системе с квантованием

Кванты, выделяемые процессам, могут быть одинаковыми для всех процессов или различными. Кванты, выделяемые одному процессу, могут быть фиксированной величины или могут изменяться в разные периоды жизни процесса. Процессы, которые не полностью использовали выделенный им квант (например, из-за ухода на выполнение операций ввода-вывода), могут получить или не получить компенсацию в виде привилегий при последующем обслуживании.

Рисунок 3.6 – Квантование с предпочтением потоков, интенсивно обращающихся к вводу-выводу

Другая группа алгоритмов использует понятие «приоритет» процесса.

Приоритет – это число, характеризующее степень привилегированности процесса при использовании ресурсов вычислительной машины, в частности, процессорного времени: **чем выше приоритет, тем выше привилегии**.

Чем выше привилегии процесса, тем меньше времени он будет проводить в очередях.

Существует две разновидности приоритетных алгоритмов: алгоритмы, использующие **относительные приоритеты**, и алгоритмы, использующие **абсолютные приоритеты**.

В обоих случаях выбор процесса на выполнение из очереди готовых осуществляется одинаково: выбирается процесс, имеющий **наивысший приоритет**.

По-разному решается проблема определения **момента смены** активного процесса. В системах с относительными приоритетами активный процесс выполняется до тех пор, пока он сам не покинет процессор, перейдя в состояние ОЖИДАНИЕ (или же произойдет ошибка, или процесс завершится).

Рисунок 3.8 – Граф состояний потока в системе с относительными приоритетами

В системах с **абсолютными приоритетами** выполнение активного процесса прерывается еще при одном условии: если в очереди готовых процессов появился процесс, приоритет которого выше приоритета активного процесса. В этом случае прерванный процесс переходит в состояние готовности.

Во многих операционных системах алгоритмы планирования построены с использованием как квантования, так и приоритетов. Например, в основе планирования лежит квантование, но величина кванта и/или порядок выбора процесса из очереди готовых определяется приоритетами процессов.

Рисунок 3.9 – Граф состояний потоков в системах с абсолютными приоритетами

В качестве примера рассмотрим схему назначения приоритетов потокам, принятую в операционной системе Windows .

В системе определено 32 уровня приоритетов и два класса потоков – потоки реального времени и потоки с переменными приоритетами.

Диапазон от 1 до 15 включительно отведен для потоков с переменными приоритетами, а от 16 до 31– для более критичных ко времени потоков реального времени (приоритет 0 зарезервирован для системных целей).

При создании процесса он в зависимости от класса получает по умолчанию базовый приоритет в верхней или нижней части диапазона.

Базовый приоритет процесса в дальнейшем может быть повышен или понижен операционной системой.

Пусть, например, значение базового приоритета некоторого процесса равно *K*. Тогда все потоки данного процесса получат базовые приоритеты из диапазона [*K-2*, *K+2*]. Отсюда видно, что, изменяя базовый приоритет процесса, ОС может влиять на базовые приоритеты его потоков.

ОС может повышать приоритет потока (который в этом случае называется динамическим) в тех случаях, когда поток не полностью использовал отведенный ему квант, или понижать приоритет, если квант был использован полностью.

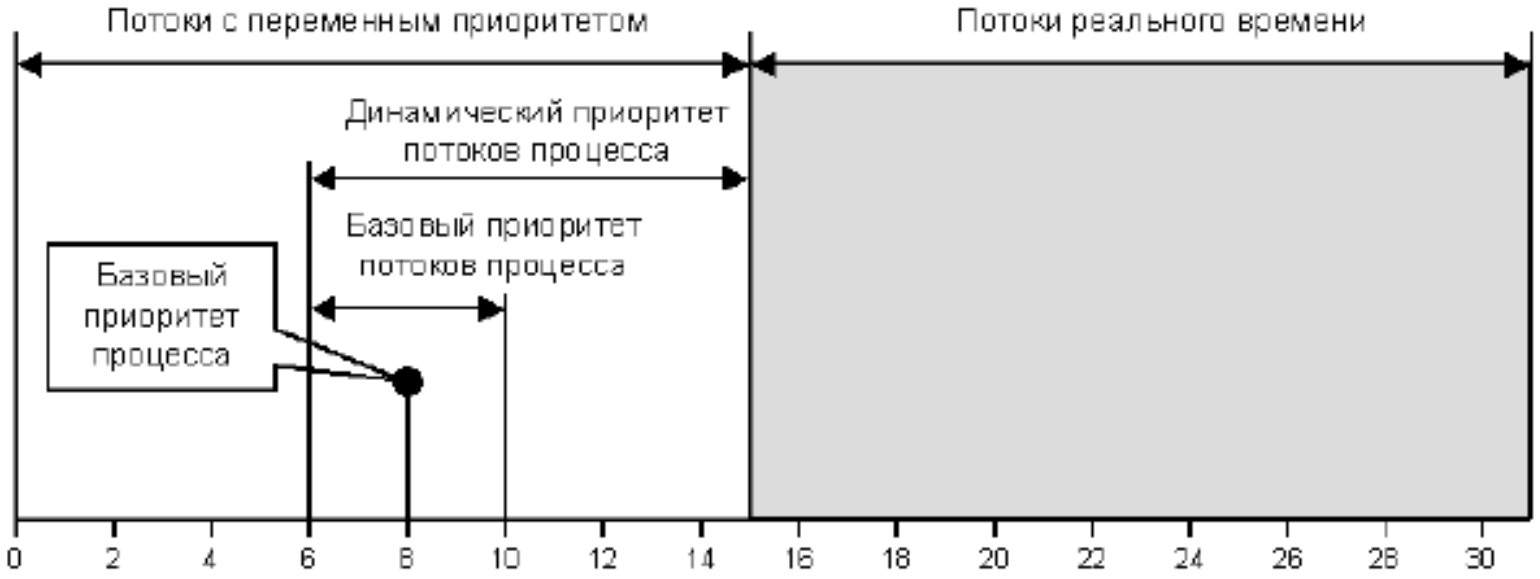
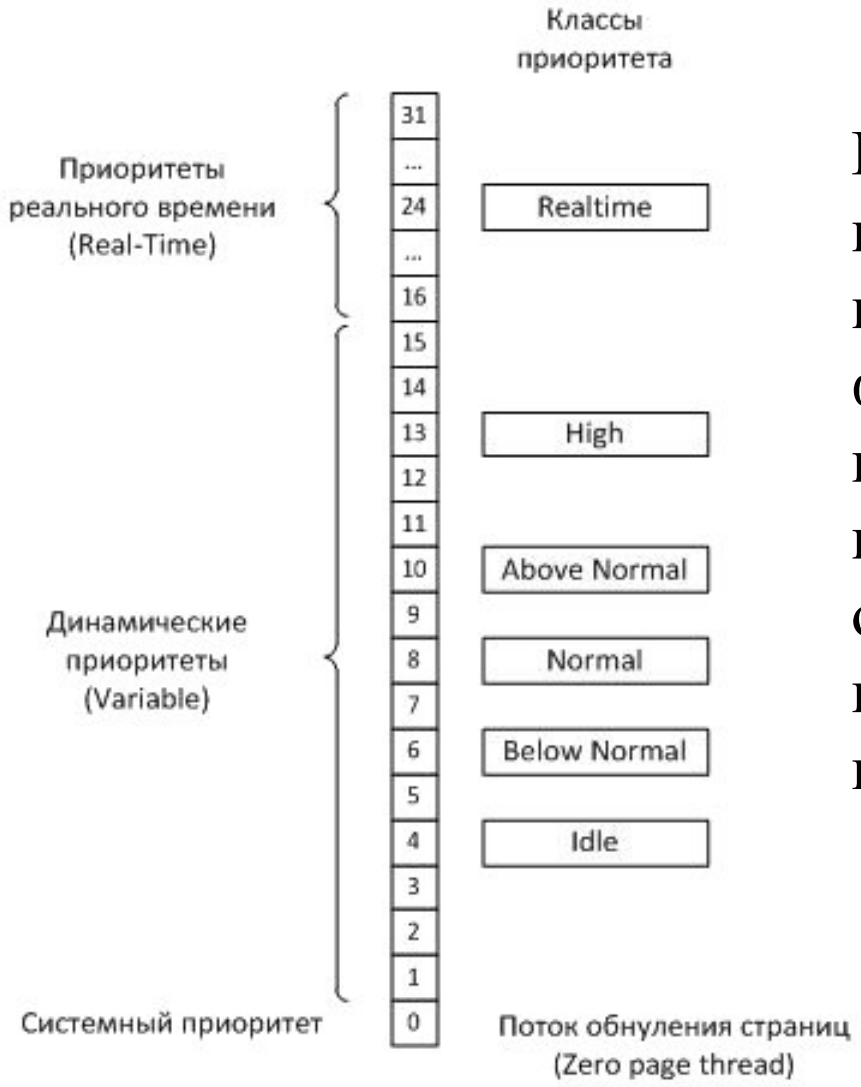


Рисунок 3.7 – Схема назначения приоритетов в Windows NT



Приоритеты от 16 до 31 используются для выполнения основных функций ОС и обычно не применяются для приложений; а приоритеты от 1 до 15 - определяют процессорный приоритет приложения.

В Windows реализован смешанный алгоритм планирования – вытесняющий, на основе квантования и приоритетов.

В Windows отсутствует единый модуль, отвечающий за планирование потоков. Алгоритм планирования реализуется несколькими процедурами ядра, совокупность которых называется **диспетчером ядра** (kernel's dispatcher).

1. Выбор потока на выполнение.

Просматривается очередь готовых к выполнению потоков и выбирается первый поток в очереди с наибольшим приоритетом, которому для выполнения предоставляется квант времени.

2. Переход выполняющегося потока в состояние ожидания.

Выполняющийся поток вызывает одну из функций ожидания и освобождает процессор. Его квант времени не истек и сохраняется за потоком, но при выходе из состояния ожидания уменьшается на единицу. Диспетчер ядра выбирает на выполнение первый поток из очереди с наибольшим приоритетом.

3. Вытеснение потоком с большим приоритетом.

Во время выполнения поток может быть вытеснен при появлении потока с большим приоритетом. Такая ситуация может возникнуть по следующим причинам:

- поток с большим приоритетом завершил ожидание;
- приоритет потока в очереди готовности динамически увеличился;
- в системе создан поток с большим приоритетом.

В любом случае выполняющийся поток вытесняется, помещается в начало очереди готовности с соответствующим приоритетом; при этом неистраченная часть кванта остается за потоком.

4. Завершение кванта времени

Когда квант времени, предоставленный потоку, истекает, операционная система проверяет, есть ли в очереди готовности поток с таким же приоритетом или выше. Если есть, то поток помещается в конец соответствующей очереди готовности и новый поток выбирается на выполнение. Если такие потоки отсутствуют, выполняющемуся потоку может быть предоставлен новый квант времени.

Динамическое повышение приоритета

Если бы операционная система осуществляла планирование потоков только на основе выше рассмотренных ситуаций, большинство потоков с низким приоритетом вообще никогда не выполнялись бы – диспетчер ядра все время выбирал бы потоки с наивысшим приоритетом.

Чтобы дать всем потокам шанс на выполнение ОС применяет **механизм динамического повышения приоритета** (Priority Boosts), который работает в следующих случаях:

- возникает событие диспетчера ядра;
- завершается операции ввода/вывода;
- происходит событие пользовательского интерфейса;
- поток слишком долго ожидает ресурс;
- поток слишком долго ожидает своей очереди на выполнение.

Никогда не повышаются приоритеты потоков реального времени (16–31).

Вопросы к лекции

3.3.1 Дайте определение процесса и потока.

3.3.2 Как происходит создание и завершение процессов и потоков?

3.3.3 В каких состояниях может находиться поток?

3.3.4 Как осуществляется планирование процессов?

3.3.5 Чем отличаются алгоритмы планирования, основанные на квантовании, и алгоритмы, основанные на приоритетах?

3.3.6 Чем отличаются вытесняющие и невытесняющие алгоритмы планирования?

3.3.7 Опишите механизм планирования, реализованный в Windows.