ОПЕРАЦИОННЫЕ СИСТЕМЫ

https//vikchas.ru

Лекция 2 «Требования к современным операционным системам. Автономная ЭВМ»

Часовских Виктор Петрович
доктор технических наук, профессор
кафедры ШИиКМ, ФГБОУ ВО
«Уральский государственный
экономический университет

Екатеринбург 2026

Основные требования к операционным системам

Главным требованием, предъявляемым к ОС, является выполнение ею основных функций: эффективного управления ресурсами и обеспечения удобного интерфейса для пользователя и прикладных программ. Кроме требований функциональной полноты, к современной ОС должны предъявляться требования, которые обычно относят к классу эксплуатационных.

Расширяемость. Аппаратная часть компьютера изменяется значительно быстрее, чем меняются ОС, жизненный цикл которых обычно длится несколько лет. Яркий пример такой ОС-долгожителя — это ОС семейства UNIX.

ОС должна поддерживать появляющиеся технологии, новые типы устройств без полного переписывания ее кода.

Если код ОС написан так, что все дополнения и модификации могут вноситься без его полного переписывания, то такую ОС называют расширяемой.

Это может быть получено за счет модульной, объектно ориентированной структуры, с хорошо разработанными интерфейсами между модулями.

Переносимость. Код ОС должен легко переноситься с одной аппаратной платформы на другую.

Под аппаратной платформой понимается не только тип процессора, но и способ организации всей аппаратуры.

Иногда употребляют другой термин — много платформенность — это свойство ОС иметь несколько версий реализации для разных аппаратных платформ.

Совместимость. Существуют ОС, имеющие множество версий, часто для них употребляют термин «семейство». В течение жизненного цикла этих семейств для них создается множество различных приложений. Пользователь привыкает работать с каким-либо приложением и ему хотелось бы продолжать с ним работать и в другой версии этой же ОС.

ОС обладает свойством совместимости с другой ОС (или другой версией той же ОС), если в ее среде выполняются приложения, разработанные для этой другой ОС.

При этом различают совместимость на уровне двоичных кодов (исполняемый код сразу запускается при переносе в среду новой ОС) и совместимость на уровне исходных текстов (когда требуется ряд дополнительных действий по трансляции и компоновке приложения).

Защищенность. Современная ОС должна защищать данные и другие ресурсы вычислительной системы от основных классов угроз.

Защищенная ОС обязательно должна содержать средства

разграничения доступа пользователей к своим ресурсам, а также средства проверки подлинности пользователя, начинающего работу с ней. Кроме того, защищенная ОС должна содержать средства противодействия случайному или преднамеренному выводу ее из строя.

Надежность и отказоустойчивость. Система должна быть защищена от ошибок, сбоев и отказов. Ее реакции на любые события должны быть предсказуемыми, а приложения не должны иметь возможности наносить вред. Файловые системы такой ОС должны быть восстанавливаемыми, она должна поддерживать аппаратные средства обеспечения отказоустойчивости, такие, например, как RAID-массивы и (или) источники бесперебойного питания.

Производительность. ОС должна обладать таким максимальным быстродействием и временем реакции, какое позволяет аппаратная платформа, на которой она установлена. На производительность ОС влияет множество факторов, наиболее значимые — это ее архитектура, алгоритмы выполнения ее функций, технологии и качество программирования, возможность мультипроцессирования и т. д.

Классификация операционных систем

Существует множество классификаций ОС, основанных на различных характеристиках.

Поддержка многозадачности. По числу одновременно выполняемых задач ОС могут быть разделены на два класса:

1) однозадачные (например, MS-DOS);

2) многозадачные (все современные ОС для настольных компьютеров), которые можно разделить на 2 класса:

- с вытесняющей многозадачностью (NetWare, Windows 3.x);
- невытесняющей многозадачностью (современные версии Linux и Windows).

Основным ресурсом любой вычислительной системы является процессорное время.

Способ **разделения процессорного времени** между несколькими одновременно существующими в системе процессами (или потоками, если таковые определены в системе) во многом определяет свойства ОС. Разницей между вытесняющей и не вытесняющей многозадачностью является степень централизации планирования процессов.

При не вытесняющей многозадачности, планирование процессов целиком осуществляется ОС, а при вытесняющей — оно распределено между ОС и самими процессами.

При не вытесняющей многозадачности, активный процесс сам определяет моменты освобождения процессора и передает управление ОС, а та выбирает из очереди готовый к выполнению процесс. При вытесняющей многозадачности, все решения о моментах переключения и

процессах, которые будут загружены на процессор, принимаются ОС, а не активными процессами.

Под поддержкой многопользовательского режима чаще всего понимают возможность работы под разными регистрационными именами с различными настройками. По **числу одновременно работающих пользователей ОС** делятся:

1) на однопользовательские (MS-DOS, ранние версии OS/2);
2) многопользовательские (современные версии Linux и Windows).

Главным отличием многопользовательских систем от однопользовательских является наличие разграничения доступа (защиты) к данным пользователей.

Многопроцессорные ОС делятся на **асимметричные и симметричные**.

Критерием деления является способ организации вычислительного процесса в системе с несколькими процессорами.

Все системные процессы **асимметричной** ОС выполняются только на одном из процессоров системы, а процессы, запущенные пользователем, распределяются на остальные процессоры.

В **симметричной** ОС любой процесс может выполняться на одном из процессоров в соответствии с алгоритмом распределения, заложенным в ОС.

По **типу аппаратуры** различают ОС мобильных устройств, планшетов, настольных компьютеров, специализированного компьютерного оборудования (графические станции, мейнфреймы).

Среди перечисленных типов устройств могут быть как однопроцессорные, так и многопроцессорные конфигурации. В любом из этих случаев особенности аппаратной платформы отражаются на

специфике ОС.

Очевидно, что ОС большой машины является более сложной и функциональной, чем ОС мобильного устройства.

По критерию «особенности областей использования» (критерий эффективности) многозадачные ОС подразделяются на три типа в соответствии с использованными при их разработке критериями эффективности:

1) системы пакетной обработки (например, ОС ЕС);
2) системы разделения времени (Windows, Linux, MAC OS);
3) системы реального времени (QNX, RT/11).

Системы реального времени применяются в основном в технических системах для управления различными объектами или процессами. В таких системах важна такая характеристика, как предельно допустимое время — это максимальное время, за которое должна быть выполнена программа, осуществляющая управляющие воздействия на объект или процесс.

Критерием эффективности для систем реального времени является их способность удерживать значение данной характеристики в определенных пределах. При этом следует отметить, что не всегда удается эффективно

Функциональные компоненты операционной системы автономного компьютера

Основные задачи, решаемые ОС, следующие:

1) управление ресурсами;

2) интерфейс с пользователем.

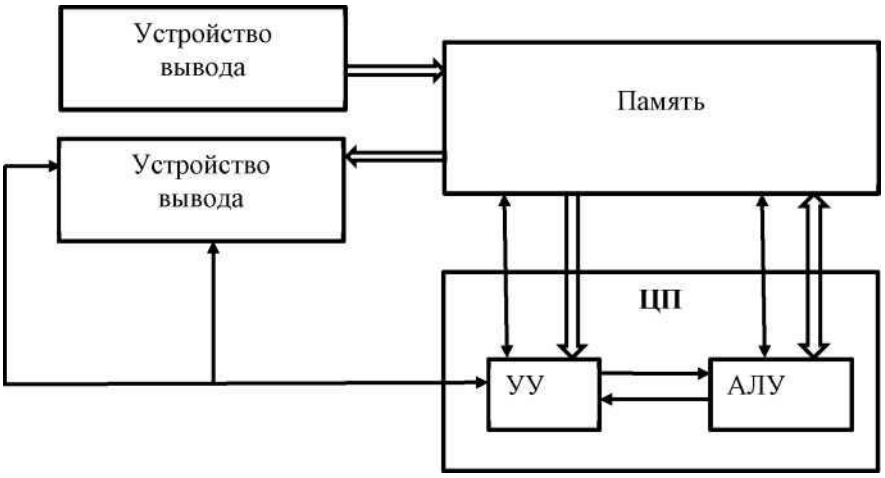
Функции ОС автономного компьютера обычно группируются по двум основным признакам:

1) в соответствии с типами локальных ресурсов;

2) со специфическими задачами, решаемыми в отношении всех ресурсов.

Такие группы функций называют подсистемами, следовательно, можно выделить подсистемы двух типов: функциональные и общие.

Рассмотрим ОС автономного компьютера (не рассматриваем сетевые функции). Для понимания того, какими ресурсами должна управлять ОС, обратимся к классической схеме устройства, являющегося вычислительной машиной архитектуры фон Неймана (рис.1).



УУ – устройство управления. АЛУ - арифметико-логическим устройством

Рис. 1. Схема вычислительной машины архитектуры фон Неймана

Машина фон Неймана образована запоминающим устройством (ЗУ),

(АЛУ), устройством управления (УУ) и устройствами ввода-вывода.

В современных компьютерах функции АЛУ и УУ выполняет единое устройство — центральный процессор (ЦП).

На рисунке толстыми (двойными) стрелками показаны потоки команд и данных, а обычными — передача между отдельными устройствами компьютера управляющих и информационных сигналов.

Основными ресурсами являются: время центрального процессора, память (основная, или оперативная, память) и устройства ввода-вывода.

Следовательно, основными **функциональными подсистемами** являются подсистемы управления временем ЦП (процессами), основной памятью и устройствами ввода-вывода.

К **общим подсистемам** обычно относят: подсистему интерфейса с пользователем, подсистему администрирования и подсистему безопасности. Структура ОС как системы показана на рис. 2.

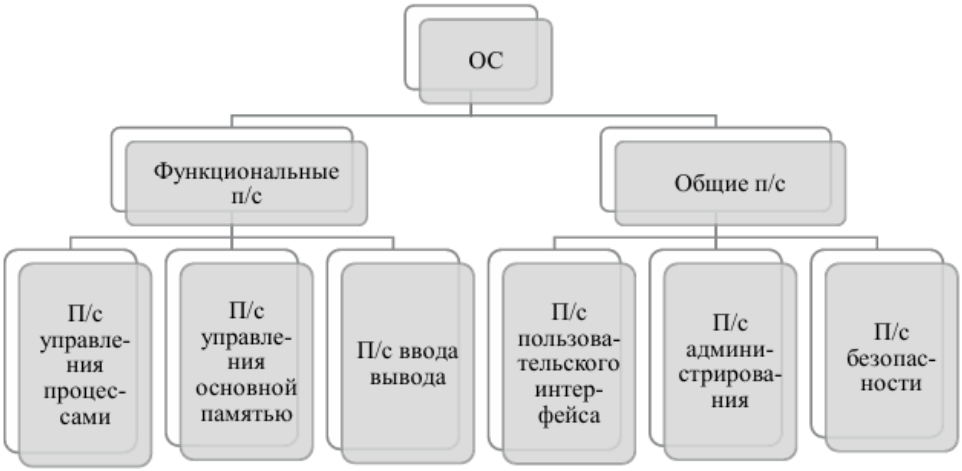


Рис.2. Структура ОС

Подсистема управления процессами

Подсистема управления процессами является одной из основных в структуре ОС.

Под **процессом** традиционно понимают программу в стадии выполнения.

Для каждого вновь создаваемого процесса ОС создает системные информационные структуры, которые содержат как данные о потребностях процесса в ресурсах системы, так и актуальные данные о фактически выделенных ему ресурсах.

Поэтому процесс часто определяют как некоторую заявку на системные ресурсы. К этим ресурсам относятся:

1) область основной памяти, в которой будут размещены коды и данные процесса (эта область называется адресным пространством процесса);
2) процессорное время;
3) файлы и устройства ввода-вывода.

Поскольку все современные ОС являются мультипрограммными, основными задачами, которые должна выполнять подсистема управления процессами, являются:

1) создание и уничтожение процессов (т. е. информационных структур, связанных с процессами);
2) выделение и учет ресурсов, выделенных каждому процессу;
3) защита ресурсов, выделенных одному процессу, от доступа к ним остальных процессов;

4) организация и обслуживание очередей заявок процессов на ресурсы (очереди к ЦП, к устройству печати и т. д.);
5) переключение с процесса на процесс;
6) организация взаимодействия процессов.

При выполнении этих функций, подсистема управления процессами обращается к другим функциональным подсистемам ОС, таким как подсистема управления основной памятью и подсистема ввода-вывода.

Примерами информационных структур, содержащих сведения о процессах, являются: блок управления задачей (TCB — Task Control Block) в OS/360, управляющий блок процесса (PCB — Process Control Block) в QS/2, дескриптор процесса в UNIX (Linux), **объект-процесс** (process object) в Windows.

В информационные структуры процесса обычно входит следующее:
1) системный идентификатор процесса (в системах Linux — дополнительно идентификатор родительского процесса);
2) системный идентификатор пользователя, создавшего процесс;
3) данные о расположении в памяти кодов и данных процесса;
4) степень привилегированности процесса
5) и др.

Часто также включаются вспомогательные данные, характеризующие историю пребывания процесса в системе, такие как:
1) доля времени, потраченная процессом на операции ввода-вывода, и доля, потраченная на вычисления;
2) текущее состояние процесса.

Данные подобного типа учитываются ОС при принятии решения о моменте и объеме предоставления ресурсов процессу.

В мультипрограммной ОС одновременно может существовать несколько процессов.

Некоторые из них порождаются по инициативе пользователей и их приложений, эти процессы называют **пользовательскими**.

Другие процессы инициализируются ОС для выполнения своих функций, они называются **системными**.

Во время своего жизненного цикла процесс может быть многократно прерван и восстановлен на процессор.

Для того чтобы возобновить выполнение процесса, необходимо восстановить состояние его операционной среды. Состояние операционной среды процесса идентифицируется:

1) по состоянию регистров;
2) состоянию программного счетчика;
3) режиму работы процессора;
4) указателям на открытые файлы;
5) информации о незавершенных операциях ввода-вывода;
6) кодам ошибок выполняемых данным процессом системных вызовов.

Эту информацию определяют как **контекст процесса**. В таком случае говорят, что при смене процесса происходит переключение контекстов.

ОС выполняет также функции по **синхронизации процессов**, которые обеспечивают приостановку процесса до наступления определенного события в системе, например завершения операции ввода (или вывода), которую выполняет ОС по запросу этого процесса.

Когда процесс завершается, ОС удаляет всю информацию о его

пребывании в системе: закрывает все файлы, которые читал и в которые записывал процесс, освобождает области основной памяти, отведенные для размещения его кодов и данных. Выполняется коррекция системных очередей и списков ресурсов, ссылающихся на завершенный процесс.

Понятие «процесс» и «поток»

В некоторых современных ОС кроме понятия «процесс» существует понятие «поток» (thread).

Поток — это средство, с помощью которого можно распараллелить процесс, иными словами, поток — это некая сущность внутри процесса, которой выделяется процессорное время для выполнения.

Существует множество процессов, в которых определенные действия можно выполнять одновременно (например, в текстовом процессоре какой-то текст можно редактировать, а какой-то — выводить на печать), эти действия и оформляются в виде отдельных потоков команд.

Можно было бы оформлять их как отдельный процесс, но это не всегда целесообразно в связи с накладными расходами, которые требует процесс (например, на создание достаточно сложных и больших по размеру занимаемой памяти информационных структур), также часто необходимо разделять ресурсы, которые при различных процессах будут изолированы друг от друга.

В каждом процессе существует хотя бы один поток.

Первичный поток создается системой сразу при создании процесса. Далее этот поток может породить другие потоки, те в свою очередь новые и т. д. Таким образом, один процесс может содержать несколько потоков, а они исполняют код в адресном пространстве процесса.

В ОС, где одновременно существуют процессы и потоки, **процесс** —

это заявка на потребление всех видов ресурсов, кроме единственного — времени процессора. Этот ресурс распределяется ОС между другими единицами работы — потоками.

Преимущества введения понятия потока:

1) создание потоков требует от ОС меньше накладных расходов, чем при создании процессов;
2) мультипрограммирование на уровне потоков повышает производительность системы;
3) использование потоков позволяет создавать более читабельные и логичные программы.

Создание процессов и потоков в ОС Windows

Создание процесса Windows осуществляется вызовом такой функции, как CreateProcess () (или ее аналогов), и проходит в несколько этапов:

1) открывается исполняемый файл, который будет выполняться в процессе;
2) создается объект Win32 «процесс»;
3) создается первичный поток;
4) подсистеме Win32 посылается сообщение о создании процесса и потока;
5) начинается выполнение созданного первичного потока;
6) в контексте процесса и первичного потока инициализируется адресное пространство (загружаются библиотеки) — программа начинает выполняться.

Пример создания процесса, в котором выполняется приложение

«Калькулятор»:

```
#include <windows.h>
int main(int argc, char* argv[])
{
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));
    if(!CreateProcess(NULL, "c:/windows/calc.exe", NULL,
NULL, FALSE,0, NULL, NULL, &si, &pi)) return 0; //Закрыть
структуры, определяющие процесс и поток
    CloseHandle(pi.hProcess);
CloseHandle(pi.hThread); return 0;
}
```

Процесс завершается:

1) если входная функция первичного потока возвратила управление;
2) один из потоков процесса вызвал функцию ExitProcess();
3) поток другого процесса вызвал функцию TerminateProcess().

Планирование и диспетчеризация потоков

Как обсуждалось ранее в отношении процессов, потоки во время своего жизненного цикла могут быть многократно прерваны и возобновлены.

В системе, не поддерживающей понятие потока, все изложенное далее может быть отнесено к процессу.

Переход от выполнения одного потока к другому осуществляется в результате планирования и диспетчеризации.

Активность по определению того, в какой момент прервать выполнение текущего активного потока и какой поток выбрать на выполнение, называется **планированием.** Планирование потоков осуществляется на основе данных, хранящихся в информационных структурах потоков.

При планировании может приниматься во внимание:

1) приоритет потоков;
2) время их ожидания в очереди;
3) накопленное время выполнения;
4) интенсивность обращений к вводу-выводу
5) и другие факторы.

Существует множество различных алгоритмов планирования потоков,

по-своему решающих каждую из приведенных выше задач.

Алгоритмы планирования могут иметь разные цели.

Например, в одном случае может быть выбран алгоритм планирования, который гарантирует, что ни один поток не будет занимать процессор дольше определенного значения, в другом случае целью может быть максимально быстрое выполнение небольших по времени задач, а в третьем случае процессорное время в первую очередь будут получать потоки интерактивных приложений.

Именно алгоритмы планирования заложены в основу одной из наиболее распространенных классификаций ОС, когда система относится либо к системам пакетной обработки, либо к системам разделения времени, либо к системам реального времени

У ОС есть планировщик статического типа, если он принимает решения заранее, а не во время работы системы.

Результатом работы такого планировщика является расписание — таблица, в которой указывается, какому потоку, когда и в течение какого временного интервала должен быть предоставлен процессор. После того как расписание создано, оно применяется ОС для переключения потоков и процессов.

Динамический планировщик принимает решения не заранее, а во время работы компьютера, основываясь на сложившейся в нем ситуации. ОС, имеющая статического планировщика, на составление расписания затрачивает ресурсов меньше, чем при динамическом планировщике, дальнейшие действия такой ОС сводятся в основном к диспетчеризации потоков или процессов.

Диспетчеризация является реализацией найденного в результате планирования решения. Если планирование можно считать стратегией, то

диспетчеризация — это тактика, реализованная через поведение системы.

Прежде чем прервать выполнение потока, ОС запоминает его контекст, т.е. ту информацию, которая будет необходима для возобновления его выполнения. Контекст отражает как состояние аппаратуры компьютера в момент прерывания, так и параметры операционной среды: коды ошибок системных вызовов, ссылки на открытые файлы, данные об операциях ввода-вывода и др.

Диспетчеризация сводится к следующему:

1) сохранение контекста текущего потока, который требуется вытеснить с процессора;
2) загрузка контекста нового потока, который был выбран планировщиком;
3) запуск выбранного потока на выполнение.

Поскольку операция переключения контекстов оказывает значительное влияние на производительность компьютера, диспетчеризация реализуется совместно с аппаратными средствами процессора (эти средства можно считать частью ОС, поэтому ОС определяют не просто как совокупность взаимосвязанных программных модулей, а как программно-аппаратный комплекс).

Состояния потока

ОС выполняет планирование потоков на основе анализа их состояний. В мультипрограммной системе поток может находиться в одном из трех основных состояний:

1) выполнения — активное состояние потока, во время которого поток обладает всеми необходимыми ресурсами и непосредственно выполняется процессором;

2) ожидания — пассивное состояние потока, находясь в котором, поток заблокирован по своим внутренним причинам (ждет осуществления некоторого события, например, завершения операции ввода-вывода, получения сообщения от другого потока или освобождения какого-либо необходимого ему ресурса);

3) готовности — также пассивное состояние потока, но в этом случае поток заблокирован из-за внешнего по отношению к нему обстоятельства (имеет все требуемые для него ресурсы, готов выполняться, однако процессор занят выполнением другого потока).

Состояния выполнения и ожидания могут быть отнесены и к задачам, выполняющимся в однопрограммном режиме, а вот состояние готовности характерно только для режима мультипрограммирования.

Жизненный цикл любого потока есть смена его состояний в соответствии с алгоритмом планирования, принятым в данной ОС.

Рассмотрим граф, отражающий смену состояний потока рис. 3.

Каждый вновь созданный поток находится в состоянии готовности, он готов в любой момент начать выполняться, для чего стоит в очереди из таких же готовых к выполнению потоков к процессору.

Когда в результате планирования подсистема управления потоками принимает решение о выполнении данного потока, тот переходит в состояние выполнения и пребывает в нем до тех пор, пока сам добровольно не освободит процессор или пока не будет вытеснен с процессора по окончании выделенного ему кванта времени.

Поток освобождает процессор добровольно, если ожидает наступления какого-нибудь события, например окончания ввода-вывода на внешнее устройство, при этом он переходит в состояние ожидания.

В случае принудительного вытеснения с процессора, поток все еще готов выполняться, поэтому переходит в состояние готовности. В состояние готовности поток переходит также из состояния ожидания, когда это ожидаемое событие наступило.

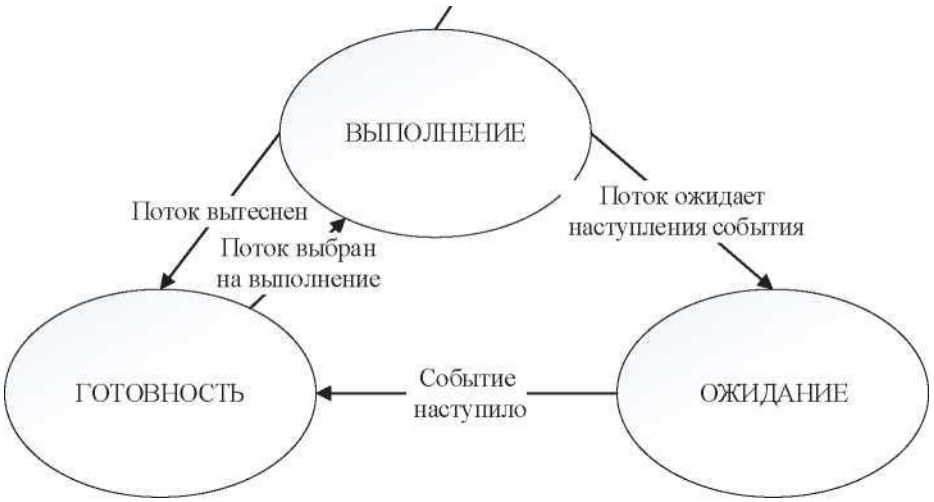


Рис. 3. Граф смены состояний потока

В состоянии выполнения в однопроцессорной системе может находиться не более одного потока, а в состоянии ожидания и готовности — несколько потоков.

Эти потоки образуют очереди, программно это реализуется через организацию списковых структур из описателей таких потоков. Списковые структуры подразумевают, что каждый описатель потока содержит по крайней мере один указатель на другой описатель, соседствующий с ним в очереди, (односвязные списки) или два указателя — на предшествующий и последующий элемент (двусвязные списки). Такая организация очередей позволяет легко их переупорядочивать, включать и исключать из них потоки.

Алгоритмы планирования

С самых общих позиций все множество алгоритмов планирования можно разделить на два класса: вытесняющие и не вытесняющие. Не вытесняющие *(non-preemptive)* алгоритмы основаны на том, что активному потоку позволяется выполняться, пока он сам, по собственной инициативе, не отдаст управление ОС для того, чтобы та выбрала из очереди другой готовый к выполнению поток.

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

Степень централизации механизма планирования потоков лежит в основе различий между вытесняющими и не вытесняющими алгоритмами. Используя не вытесняющие алгоритмы, можно создать достаточно эффективную систему, т. к. переключение с потока на поток возможно производить в те моменты времени, когда для этого требуется меньше

системных ресурсов.

Примером достаточно успешного использования не вытесняющего планирования являлись версии ОС Netware, в которых высокая скорость выполнения файловых операций была достигнута именно благодаря такому планированию. Однако недостатком таких систем является то, что ошибка в кодировании некоторого потока может привести к тому, что он не будет добровольно уступать процессорный ресурс, а удалить его принудительно в такой системе практически невозможно. Это приводит к «зависанию» подобных систем. Поэтому почти во всех современных ОС реализованы вытесняющие алгоритмы планирования потоков (процессов).

В основе многих вытесняющих алгоритмов планирования лежит концепция квантования (рис. 3.2), которая предполагает, что каждому потоку, находящемуся в системе, поочередно для выполнения предоставляется ограниченный непрерывный период процессорного времени, называемый квантом. Смена активного потока происходит:

1) если поток завершился и покинул систему;
2) произошла ошибка;
3) поток перешел в состояние ожидания;
4) исчерпан квант процессорного времени, отведенный данному потоку.

Поток, исчерпавший свой квант, переходит в состояние готовности и ожидает в очереди готовых потоков предоставления ему нового кванта процессорного времени, а планировщик выбирает новый поток из очереди готовых к выполнению в соответствии с заложенным алгоритмом. Кванты времени, выделяемые потокам, могут быть равными для всех потоков или могут различаться. На рис. 4 показан вариант планирования, при котором потокам предоставляются кванты одинаковой длины *q*. При этом можно приблизительно оценить время ожидания потока в очереди, оно равно *q*(*n*—1).

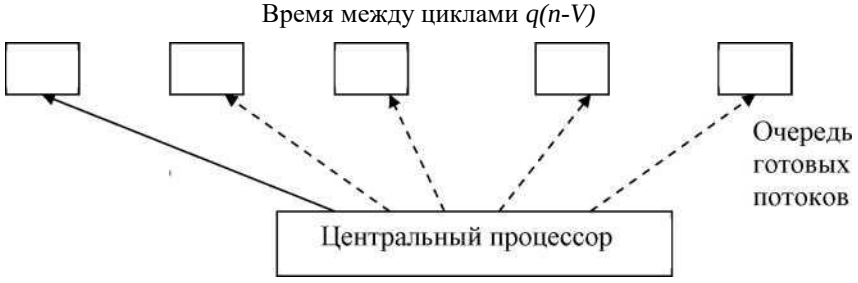


Рис. 4. Схема планирования, основанная на квантовании

Примечание. В алгоритмах, основанных на квантовании, какую бы цель они не преследовали (предпочтение коротких или длинных задач, компенсация недоиспользованного кванта или минимизация накладных расходов, связанных с переключениями), не используется никакой предварительной информации о задачах. При

поступлении задачи на обработку, ОС не имеет предварительных сведений о длительности выполнения задачи, интенсивности ее запросов к внешним устройствам, требованиях к скорости ее выполнения и т. д. Правила обслуживания при квантовании основываются на предыстории нахождения потока в системе.

Алгоритмы планирования, основанные на приоритетах

Еще одним важным понятием, которое использует большинство вытесняющих алгоритмов планирования, является понятие приоритета.

Приоритет — это число, характеризующее степень привилегированности потока при использовании ресурсов компьютера. Это важно в отношении процессорного времени: чем выше приоритет, тем более этот поток значим для системы, получает привилегии по отношению к другим потокам и тем меньше он будет находиться в очередях, быстрее будет выполнен.

Приоритет обычно выражается целым отрицательным или положительным числом.

В некоторых ОС существует правило, что приоритет потока тем выше, чем больше число, определяющее приоритет, в других системах — наоборот.

В большинстве ОС, в которых существуют потоки, приоритет потока зависит от приоритета процесса, в котором выполняется данный поток.

Приоритет процесса назначает ОС при его создании. Значение приоритета содержится в описателе процесса и учитывается при установлении приоритетов потокам этого процесса.

Во многих ОС существует возможность изменения приоритетов во время жизни потока.

Приоритет может изменить сам поток, для этого он обращается с соответствующим вызовом к ОС; или пользователь, для этого он выдает соответствующую команду. Также сама ОС может изменить приоритеты потоков в зависимости от того, какая ситуация складывается в системе. В этом случае приоритеты называются динамическими, в отличие от фиксированных (статических) приоритетов.

Система приоритетов в ОС Windows

В ОС Windows существует 32 уровня приоритетов и два класса потоков — это 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 — системный уровень, зарезервированный для потока обнуления страниц (имеются в виду страницы основной памяти).

При создании процесса, ему назначается базовый приоритет — один из шести классов

приоритетов:

1) Real time class (значение 24);
2) High class (значение 13);
3) Above normal class (значение 10);
4) Normal class (значение 8);
5) Below normal class (значение 6);
6) Idle class (значение 4).

В дальнейшем этот приоритет может быть изменен.

Приоритет каждого потока (базовый приоритет потока) складывается из приоритета его процесса и относительного приоритета самого потока.

Есть семь относительных приоритетов потоков.

1) Normal: такой же, как и у процесса;
2) Above normal: +1 к приоритету процесса;
3) Below normal: —1;
4) Highest: +2;
5) Lowest: —2;
6) Time critical: устанавливает базовый приоритет потока для Real time класса в 31, для остальных классов в 15.
7) Idle: устанавливает базовый приоритет потока для Real time класса в 16, для остальных классов в 1.

Очевидно, что, изменяя базовый приоритет процесса, ОС может изменять и базовые приоритеты его потоков. ОС также может изменять и относительный приоритет потока в зависимости от истории его существования в системе.