Конспект лекций по операционным системам.

## Базовые определения

Прикладное ПО - это программы, выполняющие задачи, требуемые пользователю
Системное ПО - это программы, способствующие функционированию прикладных программ и упрощающие их разработку
Системное ПО - это операционные системы, драйверы и фреймворки
Драйвер - программа, управляющая работой периферийного устройства
Операционная система - это программа, обеспечивающая среду выполнения для других программ и облегчающая им доступ к устройствам, составляющим компьютер, процессору, жёсткому диску и т.д.
Фреймворк - это программная среда специального назначения, используемая для того, чтобы упростить написание программ и облегчить объединение отдельных программных компонентов.

## Услуги, предоставляемые ОС

Упрощают использование аппаратных средств. Создаваемая ими виртуальная машина заметно отличается от реальной. ОС изолируют пользователей от аппаратной части компьютеров.
ОС обеспечивает распределение вычислительных ресурсов между программами.
Управляет файлами и папками. Для упрощения работы пользователей создаётся файловая система.
Предоставляет пользователю интерфейс для взаимодействия с программами и компьютером.

## Интерфейс

Интерфейс - набор средств, используемые для взаимодействия двух систем.

## Типы интерфейсов

Графический интерфейс. Управление осуществляется путём нажатия на различные виды кнопок, которые изображены на экране.
Интерфейс командной строки. Управление осуществляется путём набора текстовых команд.
Программный интерфейс. Через программный интерфейс программы взаимодействуют друг с другом. API - Application Programming Interface.
Голосовой интерфейс - команды даются с помощью голоса, речи.
Жестовый интерес - управление с помощью жестов: сенсорный экран, тачпад, джойстик, руль

Нейрокомпьютерный интерфейс - обмен данными между человеческим мозгом и электронным устройством осуществляется с помощью биологической обратной связи и встроенных электронных имплантов.
Аппаратный - предназначений для взаимодействия физических устройств друг с другом: тип разъёмов и параметры сигналов передаваемых через эти разъёмы.

# Задание. Процесс.

Задание - совокупности программы и входных данных, необходимых для её выполнения.
Процесс - экземпляр программы во время выполнения, независимый объект, которому выделены системные ресурсы, например, процессорное время и память. Каждый процесс выполняется в отдельном адресном пространстве. Один процесс не может получить доступ к данным другого процесса.
Понятие процесса включает:
1. Множество внешней по отношению к процессу информации, используемой ОС для управления ресурсом типа «процесс». Состав данной информации зависит от ОС.
2. Структура и содержимое адресного пространства процесса. Т.е. части памяти, выделенной процессу.
3. Множество ресурсов, принадлежащих процессу или используемых процессом, а также состояние этих ресурсов.

# Архитектура ОС

## 1.1 Монолитная ОС

В монолитной ОС система организуется как набор процедур, каждую из которых может вызывать пользовательская программа. Вся ОС расположена в едином адресном пространстве.
Пользовательский режим - процесс не может получить доступ к чужой области памяти, не все инструкции процессора ему доступны. Поток, исполняющийся в пользовательском режиме, может получить доступ к системным ресурсам только посредством вызова системных сервисов. Когда программа пользовательского режима вызывает системный сервис, вызов перехватывается и вызывающий процесс переключается в режим ядра.
Режим ядра - привилегированный режим работы процессора, в котором исполняется код ОС. Поток, исполняющийся в режиме ядра, имеет доступ ко всей памяти и аппаратуре.
Системные сервисы - набор программ, которые перехватывают обращение прикладных программ к системным ресурсам. Когда выполнение системного сервиса завершается, ОС переключает поток обратно в пользовательский режим.
Привилегия - это свойство, устанавливаемое при проектировании системы, которое определяет, какие компьютерные операции разрешены, какие доступы к памяти законны. Привилегии испльзуются для обеспечения без-

опасности в компьютерной системе и повышения надёжности её работы.
Привилегии реализуются путём присвоения процессам значения от 0 до 3. Значение 0 соответствует наибольшим привилегиям, тогда как значение 3 - наименьшим.
Привилегии реализуются на уровне процессора.

Достоинства монолитной ОС

1. Высокая скорость
2. Относительно простая разработка ОС:

Недостатки:

1. Поскольку всё ядро работает в одном адресном пространстве, сбой в одном из компонентов может нарушить работоспособность всей системы.
2. Усовершенствование ОС затруднено, т.к. внесение изменений в одну часть ОС может потребовать внесение изменений в другие части ОС.
Примеры монолитных ОС: MS DOS, первые версии MacOS

## 1.2 Модульная операционная система

Модульная ОС - это ОС, в которой каждый программный модуль (часть ОС) имеет законченное функциональное назначение с оговорёнными правилами взаимодействия. Все модули ОС равнозначны. В отличие от монолитной ОС, каждому модулю выделена своя область памяти. Все модули исполняются в режиме ядра.

Достоинства модульной ОС:

1. Упрощается усовершенствование ОС
2. Достаточно высокая надёжность ОС, т.к. сбой в одном модуле не влияет на другие

Недостатки модульной ОС:

1. Усложняется создание ОС
Модульную архитектуру имеет разновидность UNIX - FreeBSD. Linux имеет монолитно-модульную архитектуру.

## 1.3 Послойная ОС

Достоинства:
1. упрощается усовершенствование ОС
2. Достаточно высокая надёжность ОС
Недостатки:
1. Уменьшение быстродействия ОС.
Частично послойную архитектуру имеет ОС Windows.

## 1.4 Микроядерные ОС

В сложном программном продукте в среднем содержится 10 ошибок на 1000 строк кода. Следовательно, монолитная операционная ОС, состоящая из 5000000 строк кода, скорее всего, содержит от 10000 до 50000 ошибок. Таким образом, для уменьшения количества ошибок надо уменьшать размер кода, работающего в привилегированном режиме.
MacOS X использует микроядерную архитектуру, которая основана на микроядре Mach. При этом используются некоторые модули, взятые из ОС FreeBSD.
Harmony OS - микроядерная ОС, устанавливается на смартфоны компании Huawei.
Микроядро MINIX 3 занимает всего лишь около 12000 строк кода на языке С и 1400 строк кода на ассемблере.

Достоинства микроядерной ОС:

1. Достаточно высокое быстродействие
2. Высокая надёжность. Микроядерную архитектуру используют ОС, работающие в реальном масштабе времени в промышленных устройствах, авиации и военной технике.

Недостатки микроядерной ОС:

1. Сложность разработки
2. При увеличении числа процессов значительно падает быстродействие, т.к. увеличивается число обращений к ядру.

## 1.5 Клиент-серверная ОС

Клиент-серверная ОС разделяет процессы на два типа:
1. Процесс сервера, каждый из которых предоставляет какую-нибудь службу
2. Процесс клиента, который пользуется этими службами.
Для связяи клиентов с серверами используется ядро (микроядро).
Связь между клиентами и серверами организуется с помощью передачи сообщений следующим образом:
1. Клиентский процесс составляет сообщение, в котором говорится, что именно ему нужно, и отправляет его ядру или микроядру.
2. Ядро или микроядро ОС определяет, какой сервер должен ответить на сообщение, и доставляет сообщение серверу.
3. Служба выполняет определённую работу и отправляет обратно ответ.
4. Ядро возвращает клиенту результат в виде другого сообщения.

# Требования к ОС

Требования правительства США.

## 2.6 Совместимость с POSIX

Lorem Ipsum

## 2.7 Безопасность

Безопасность, то есть защита от несанкционированного доступа. Требования, выполнение которых делает систему многопользовательской.
А. Защита от несакционированного проникновения на компьютер.
Б. Защита ресурсов пользователя от других пользователей.
В. Возможность установления квот на системные ресурсы для предотвращения захвата одним пользователем всех системных ресурсов.
Г. Разграничение прав доступа пользователей к файлам и устройствам.
Д. Сохранение информации о действиях пользователей, связанных с безопасностью.

# Особенности реализации ОС Windows


Ядро (kernel) - компонент операционной системы, непосредственно взаимодействующий с процессом и памятью. Выполняет планирование и переключение потоком, обработку прерываний и исключений, взаимодействие с драйверами и мультипроцессорную синхронизацию.
В Windows также реализована идея помещать в ядро исполнительный механизм, а не политику.

## 3.1 Поток

В Windown - процесс не является испольняемой сущностью, т.е. непосредственно не выполняется процессором. Это некий объект, который характе-

ризует задачу во время испольнения.
Исполняемая сущность внутри процесса - это поток выполнения (thread of execution).
Поток - это программа, выполняющаяся в адресном пространстве процесса и имеющая доступ ко всем ресурсам процесса. Каждый процесс должен содержать не менее одного потока. Как правило, в Windows процесс содержит несколько потоков.
Поток - это способ распараллеливания вычислительных операций и экономии памяти.
В то время как процесс - это логическое представление работы, которую должна выполнить программа, поток отображает одну из многих необходимых подзадач.

## 3.2 Прерывания

Прерывания (interrupt) - это запрос, поступающий от устройства ввода-вывода, исполняющейся программы или процессора с требованием прервать выполнение текущей программы и запустить на выполнение другую программу. Прерывания бывают внешние и внутренние.
Внешние прерывания являются внешними по отношению к процессору, поэтому так и называются. Внешние прерывания бывают друх типов: прерывания ввода-вывода и аппаратные прерывания.
Прерывания ввода-вывода создают переферийные устройства, которые сообщают, что они готовы выполнить операцию ввода или вывода. Для выполнения операции ввода-вывода надо прервать выполнение текущей программы и выполнить программу управления вводом-выводом. Программа, вызываемая при возникновении прерывания называется программоnй обработки прерывания.
Аппаратные прерывания поступают в процессор от различных устройств компьютера и не связаны с вводом-выводом. Они сигнализируют о наступлении какого-нибудь события или об обнаружении сбоев в работе какого-нибудь устройства. В этом случае тоже вызывается соответсвтующая программа обработки прерывания.
Внутренние прерывания происходят внутри процессора и бывают двух типов:
Программными прерываниями считаются запросы со стороны программы на начало выполнения операции ввода-вывода или при обращении к функциям операционной системы.
Исключительные ситуации (исключения) возникают в тех случаях, когда процессор не в состоянии выполнить предусмотренное в программе действие, например, деление на ноль. В этом случае он прерывает выполнение программы и сообщает операционной системе о наличии исключительной ситуации, а также выдаёт необходимую дополнительную информацию, которая позволяет более точно определить причину её возникновения.

Исключительные ситуации бывают следующих типов:

Деление на ноль
Пошаговая работа (трассировка)

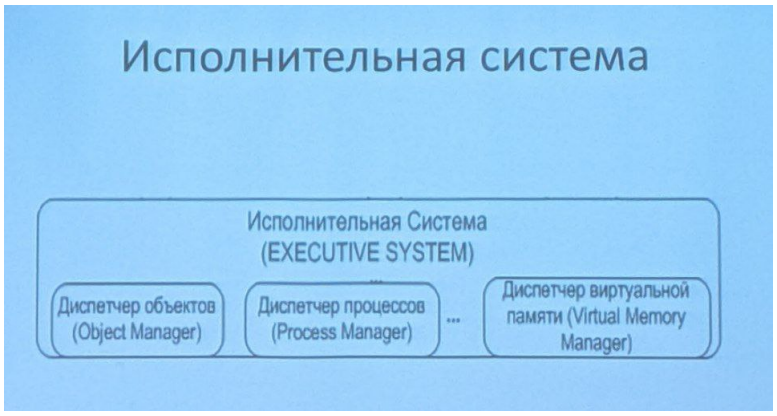
Переполнение порядка при работе со знаковым операндом
Выход индекса за границу, т.е. обращение к области памяти не принадлежащей к данной программе. В этом случае выдаётся ошибка General Protection Fault. (Нарушения основной защиты)
Другие исключительные ситуации

## 3.3 Уровень абстрагирования от оборудования

Ядро осуществляет взаимодействие с устройствами, но не непосредственно, а через уровень абстрагирования от оборудования. Чтобы работа ядра не зависела от аппаратного обеспечения введён слой абстрагирования от оборудования HAL (Hardware Abstraction Layer).
В HAL входит загружаемый модуль режима ядра (`C:\Windows\System32\hal.dll`, предоставляющий низкоуровневый интерфейс с аппаратной платформой, на которой выполняется Windows. Он скрывает от ОС специфику конкретной аппаратной платформы, т.е. все функции, зависимые от архитектуры и от конкретной машины. Таким образом, ядро работает с некоей стандартизированной виртуальной машиной. HAL преобразует команды ядра в команды, понятные конкретному оборудованию.
Также в HAL входят драйверы различных переферийных устройств.
DLL (Dynamic Link Library) - динамически подключаемая библиотека, подпрограмма, позволяющая многократное использование различными программными приложениями. К DLL относятся разные программные компоненты и в частности драйверы.
В системах UNIX также используются библиотеки. Они имеют расширение .so (shared object).

## 3.4 Исполнительная система

Исполнительная система (executive system) - спроектирована как уровень абстрагирования от ядра. Она обеспечивает специфические политики для управления объектами, памятью, процессами, файлами и устройствами. Таким образом, ядро реализует механизм, а исполнительная система - политику.
Ядро и исполнительная система включаются в один исполняемый модуль NTOSKRNL.EXE.

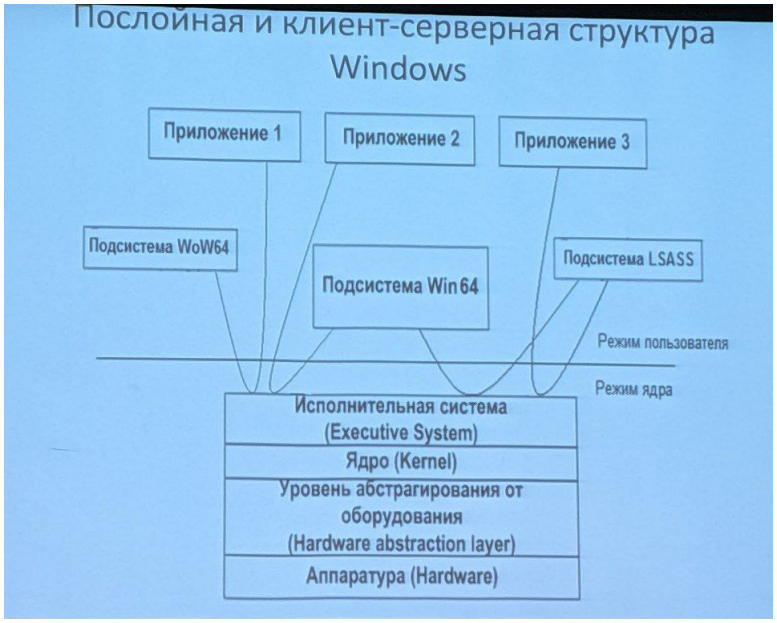


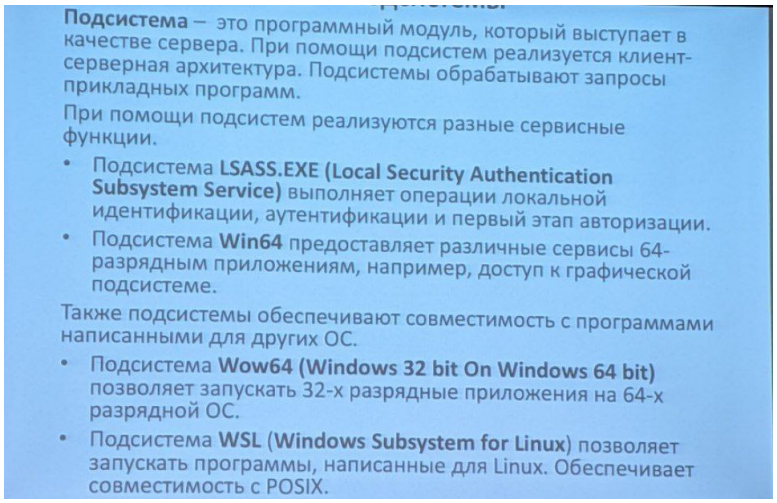

Основные модули исполнительной системы:

1. Диспетчер объектов (Object Manager).
2. Диспетчер процессов и потоков (Process and Thread Manager).
3. Диспетчер виртуальной памяти (Virtual Memory Manager).
4. Справочный монитор защиты (Security Reference Monitor).
5. Диспетчер ввода-вывода (I/O Manager).
6. Диспетчер кэша (Cache Manager).
7. Средство локального вызова процедур (Local Procedure Call).

## 3.5 Подсистема

Подсистема - это программный модуль, который выступает в качестве сервера. При помощи подсистем реализуется клиент-серверная архитектура.





**Подсистема** – это программный модуль, который выступает в качестве сервера. При помощи подсистем реализуется клиент-серверная архитектура. Подсистемы обрабатывают запросы прикладных программ.

При помощи подсистем реализуются разные сервисные функции.

- Подсистема **LSASS.EXE (Local Security Authentication Subsystem Service)** выполняет операции локальной идентификации, аутентификации и первый этап авторизации.
- Подсистема **Win64** предоставляет различные сервисы 64-разрядным приложениям, например, доступ к графической подсистеме.

Также подсистемы обеспечивают совместимость с программами написанными для других ОС.

- Подсистема **Wow64 (Windows 32 bit On Windows 64 bit)** позволяет запускать 32-х разрядные приложения на 64-х разрядной ОС.
- Подсистема **WSL (Windows Subsystem for Linux)** позволяет запускать программы, написанные для Linux. Обеспечивает совместимость с POSIX.

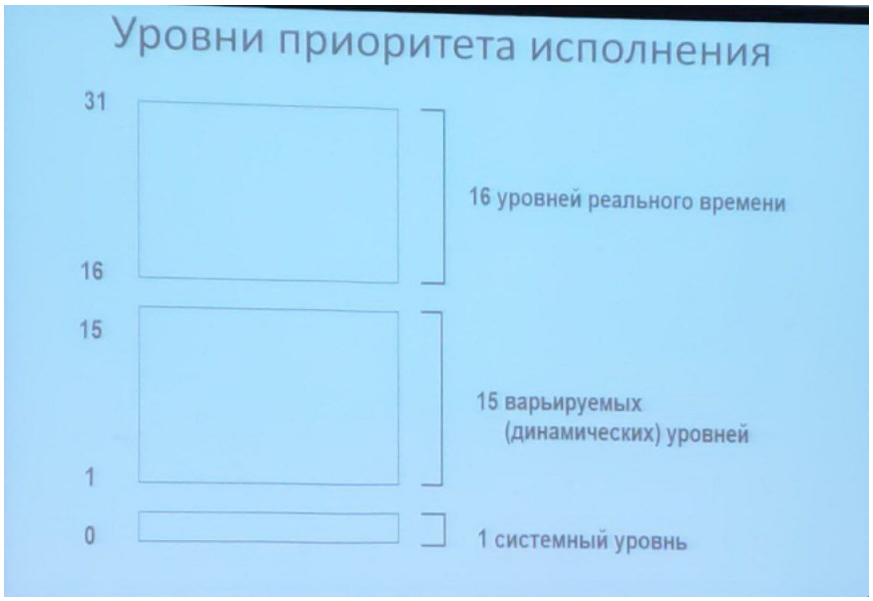
## 3.6 Приоритеты Windows

В Windows используется приоритетный, с вытеснением и кватованием времени планировщик.
Процессор выделяется потоку на квант времени, вычисляемый, как несколько тиков системных часов. Планировщик поддерживает 32 уровня приоритета и соответственно столько же различных очередей планировщика.
Алгоритм работы планировщика такой:

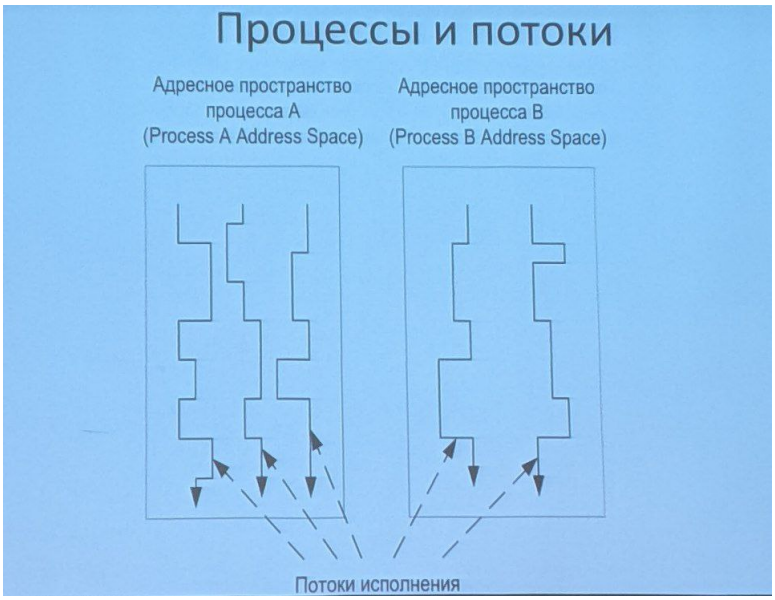
- В соответствии с приоритетом выполнения потоков создаётся несколько очередей. В каждой очереди потоки только с одинаковым приоритетом.

- Сначала выполняются потоки из очереди с самым высоким приоритетом.

- Если в этой очереди нет больше потоков, тогда планировщик будет обслуживать очередь второго по величине приоритета, потом третьего и т.д.




Самый высокий приоритет у потоков реального времени, он больше чем у потоков ОС. Такой приоритет нужен для обработки высокоскоростных потоков данных, которые нужно обрабатывать в реальном времени.
На динамическом уровне выполняются почти все прикладные системные процессы. Особенностью данного уровня является то, что приоритеты потоков могут меняться в течение времени в зависимости от ситуации и действий пользователя. Приоритет повышается для тех потоков, с которыми непосредственно в данный момент работает пользователь.
Системный уровень предназначен для одного системного процесса Idle. Этот процесс обладает самым низким приоритетом и работает, когда процессор простаивает. Процесс Idle удаляет ненужные данные из файла подкачки, также снижает энергопотребление процессора при простое. На каждом ядре процессора запускается по одному потоку процесса Idle.




Приоритеты процессов могут принимать не все возможные значения от 0 до 31, а только несколько определённых значений.

- Real Time Class (значение 14)
- High Class (значение 13)
- Above Normal class (значение 10)
- Normal Class (значение 8)
- Below Normal Class (значение 6)
- Idle Class (значение 4)

По умолчанию приоритет процесса наследуется от родительского процесса следующим образом:

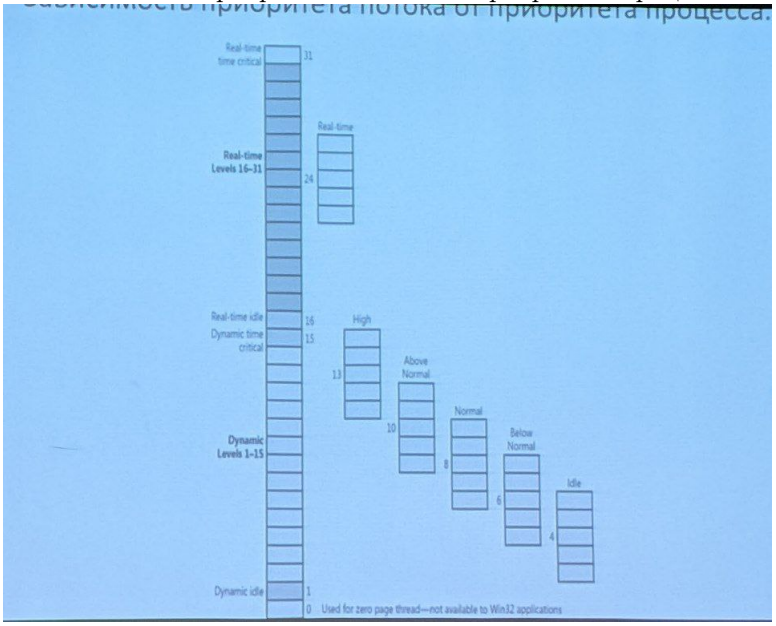
- Если приоритет родительского процесса Normal или ниже, то дочерний процесс имеет приоритет равный родительскому
- Если приоритет родительского процесса выше Normal, то дочерний процесс имеет приоритет Normal.

Приоритет потока - величина, складывающаяся из двух составных частей: приоритета породившего поток процесса и собственно приоритета потока. Приоритет потока задаётся относительно приоритета процесса.

- Normal: такой же какой и у процессора
- Above normal: +1 к приоритету процесса
- Below normal: -1;

- Highest: +2;
- Lowest: -2;
- Time critical: устанавливает базовый приоритет потока для Real Time в 31, для остальных классов в 15;
- Idle: устанавливает базовый приоритет потока для Real Time класса в 16, для остальных классов в 1.

Зависимость приоритета потока от приоритета процесса.




Многопоточность приложений нужна для следующих целей:
Распараллеливания вычислений для ускорения работы программы за счёт того, что разные потоки выполняются на разных ядрах.
Задания разным потокам разных приортетов для повышения удобства работы пользователя с программой. Таким образом базовый (начальный приоритет потока определяется процессом, но ОС может его менять для удобства работы пользвователя с программой. Когда пользователь непосредственно взаимодействует с потоком ОС повышает приоритет потока. Если пользователь перестаёт взаимодействовать с этим потоком, то приоритет понижается.

## 3.7 Изменение приоритетов

Используя системные функции Windows, программист может задать приоритет процесса при его создании и в дальнейшем при необходимости изменить.
Для создания процесса используется системная функция CreateProcess. Эта функция позволяет задать класс приоритета создаваемого процесса.

Приоритет процесса можно изменить и после его создания, используя функцию SetPriorityClass.
Команда Start позволяет создать процесс и задать его приоритет.
Пользователь получает доступ к функции SetPriorityClass, используя специальные утилиты, например, диспетчер задач и Process Explorer.
Изменение приоритетов индивидуальных потоков внутри процесса обычно не имеет смысла, так как программист определил приоритеты потоков в соответствии с логикой работы программы. Изменение относительных приоритетов потоков может привести к неадекватному поведения приложения.

## Кеш-память

Кэш-память - это промежуточная память которая сглаживает разницу в быстродействии основных видом памяти. Использование кэш-памяти значительно ускоряет обмен данными между устройствами.


Экспериментально установлено, что после обработки некоторых данных процессор с большой вероятностью обращается к тем же самым данным или к данным, находящимся в непосредственной близости от них. Это явление называется принципом пространственной локализации.
Также установлено, что программе в ближайшее время, вероятнее всего, потребуются данные, которые использовались недавно. Эта закономерность называется принципом временной локализации.
Эти два принципа используются для ускорения обмена данными с дисковыми накопителями.
При чтении данных с диска читаются не только требуемые данные, но и данные находящиеся рядом, т.е. читается не один блок данных, а несколько. Прочитанные данные автоматически заносятся в кэш. Эта процедура называется упреждающее чтение.

Перед чтением данных с жёсткого диска выполняется попытка чтения данных из кэша. Если нужные данные действительно находятся в кеше, то они с высокой скоростью записываются в основную часть оперативной памяти, не относящейся к кешу. Эта ситуация носит название попадание в кеш. Если данные в кеше отсутствуют, то говорят, что имеет место промах кеша, и данные читаются с диска.
Если кеш заполнен полность, то при записи новых данных удаляются те данные, которые записаны давно и продолжительное время