Процессорное время - время, затраченное процессором на обработку задачи. Процессорное время утилизируется в виде **вычислительных процессов** и **потоков**.

Процесс - последовательность взаимосвязанных действий, преобразующих данные.
Вычислительный процесс - последовательность операций, выполняемых компьютером для достижения определённой задачи
Он является основным объектом операционной системы, обеспечивающей управление и координацию выполнения программ.
Процесс включает в себя *ход выполнения программы* и *образ процесса (код выполнения)*, разделенный на образ *на диске (исполняемый файл)* и образ *в памяти (загруженный в оперативную память и настроенный для исполнения)*.
Образ в памяти – *сегментация.*
Основные сегменты: код, данные, стек. Могут также присутствовать сегменты BSS (неинициализированных данных), кучи и другие.

В операционных системах, таких как Windows, вычислительный процесс рассматривается как системный объект, который управляется ядром ОС. Процессы могут создавать, уничтожать и взаимодействовать друг с другом.

Атрибуты процесса:

- Идентификатор процесса (PID): уникальный идентификатор, присваиваемый каждому процессу.
- Идентификатор родительского процесса (Parent PID): указывает на процесс, создавший текущий процесс.
- Пользователь процесса, User ID (UID): идентификатор пользователя, от имени которого выполняется процесс. Часто дополняется также группой процесса, Group ID (GID).
- Права доступа: определяют, какие операции может выполнять процесс.
- Состояние: текущее состояние процесса (например, работающий, приостановленный и т.д.).
- Имя процесса: имя исполняемого файла.
- Параметры командной строки: аргументы, переданные процессу при его запуске.
- Приоритет: определяет, как часто процесс получает доступ к CPU по сравнению с другими процессами

Образ процесса — это основа, из которой создаётся процесс. Он включает в себя исполняемый *код, данные и ресурсы*, необходимые для выполнения процесса. Образ может загружаться в память и запускаться, создавая экземпляр процесса.

Адресное пространство и пространство дескрипторов процессов

- **Адресное пространство**: это диапазон адресов памяти, доступный процессу.
- **Пространство дескрипторов**: это набор дескрипторов (handles), которые представляют ресурсы, используемые процессом, такие как файлы, потоки и синхронизационные объекты. Дескрипторы позволяют процессам управлять этими ресурсами.

В течение своего существования (жизненного цикла) поток проходит ряд состояний, часть из которых «проходные», другие повторяются многократно:

Состояния процесса/потока:

- P (Passive): Пассивное состояние, поток не может выполняться и требует завершения инициализации или разрушения.
- R (Ready): Поток ожидает выделения процессорного времени, находится в очереди на выполнение.
- A (Active/Running): Поток активен и выполняется.
- W (Wait/Sleep): Поток ожидает завершения блокирующей операции.

Переходы между состояниями:
Active ↔ Ready: Планировщик осуществляет переключение.
Active → Wait: Поток выполняет блокирующий системный вызов.
Wait → Ready: Возврат из вызова, условие ожидания.

Дополнительные состояния:

- S (Suspended): Поток остановлен и может быть возобновлен другими потоками.
- Z (Zombie): Поток завершен, но его данные остаются доступны.

Состояния процесса в многопоточной среде: начальное (до начала главного потока), активное (при наличии хотя бы одного выполняющегося потока), зомби (все потоки завершены, но объект «процесс» еще сохраняется), заблокированное/тупик (все потоки остановлены или находятся в ожидании)

Многозадачность - способность ОС выполнять одновременно несколько программ. Подходы к обеспечению многозадачности:

1. **Пакетная многозадачность**: Задачи выполняются в отведенные лимиты времени, после чего удаляются.
2. **Разделение времени**: Выполнения нескольких задач на одном процессоре в виде чередующихся достаточно коротких шагов.
    2.1. **Вытесняющая многозадачность** (с квантованием времени): ОС вынуждает задачу освободить процессор в пользу другой задачи после истечения кванта времени (обычно от единиц до десятков миллисекунд).
    2.2. **Кооперативная многозадачность** (невытесняющая): задача управляет временем, которое ей предоставлено для выполнения, и должна активно освобождать процессор для других задач.
3. **Многозадачность параллельных систем**: Использование параллельных архитектур с физически отдельными исполнительными блоками.
4. **Многозадачность реального времени**: Программы основаны на обработчиках событий.

Концепция процессов оказалась недостаточной, особенно для параллельных вычислительных архитектур. Была введена новая сущность – вычислительный поток. **Вычислительный поток** - это наименьшая единица выполнения, которая может быть запущена и управляется операционной системой. Потоки позволяют программе выполнять несколько операций параллельно, улучшая производительность.

В **многопоточной** системе именно поток становится основным объектом планирования времени выполнения и участником взаимодействия параллельных программ. Процесс при этом выступает как «контейнер» потоков и обладатель вычислительных ресурсов, которые доступны всем его потокам.
две разновидности реализации многозадачности:

- «обычная», основанная на поддержке только процессов;
- многопоточная – поддерживаются также и потоки.

Процессы и потоки характеризуются контекстом, состоянием и другими атрибутами. **Контекст процесса** включает информацию, необходимую для полного описания процесса, такую как состояние регистров процессора, адресное пространство и управляющие структуры системы. Переключение между процессами осуществляется путем переключения текущего контекста, что требует времени и ресурсов из-за объема информации.

Контекст потока обычно более легкий, чем у процесса, поскольку не включает адресное пространство. Переключение контекстов потоков происходит быстрее, но если потоки принадлежат разным процессам, требуется также смена контекста процесса.

Каждый процесс может состоять из одного или нескольких потоков, которые выполняются в его контексте.

Атрибуты потока:

- Идентификатор потока (TID): уникальный идентификатор, присваиваемый каждому потоку.
- Идентификатор родительского процесса (Parent PID): указывает на процесс, в котором выполняется поток.
- Стек потока: область памяти, используемая для хранения данных, связанных с выполнением потока.
- Приоритет: уровень приоритета потока, определяющий, насколько часто он будет получать доступ к CPU.
- Состояние: текущее состояние процесса (например, работающий, приостановленный и т.д.).
- Имя потока: имя, присвоенное потоку для идентификации.

Приоритет: определяет, насколько важен поток по сравнению с другими. Приоритеты могут быть:

- Низкий: поток будет выполняться реже.
- Нормальный: стандартный уровень выполнения.
- Высокий: поток будет получать больше времени CPU.

Разделяемые адресное пространство и пространство дескрипторов потоков

- Разделяемое адресное пространство: все потоки внутри одного процесса используют одно и то же адресное пространство.
- Пространство дескрипторов: потоки могут использовать общее пространство дескрипторов, что позволяет им управлять общими ресурсами.

Диспетчер потоков отвечает за распределение времени процессора между активными потоками. Он использует алгоритмы планирования, чтобы определить, какой поток будет выполняться в данный момент.

Главный поток — это поток, который создается при запуске процесса. Он является первым потоком и управляет выполнением других потоков.

Принудительное завершение процессов и потоков нежелательно, т.к. происходит в произвольные моменты, при неизвестном их состоянии. В случае потоков возможна «утечка» ресурсов – объекты, которые были созданы потоком и не удалены при завершении, остаются в памяти до завершения всего процесса.

«Приостановленное» (**suspended**) состояние управляется внутренним счетчиком.

Локальная память потоков, Thread Local Storage (**TLS**) – позволяет иметь у разных потоков одинаковый набор переменных с разными значениями.

Нити (Fiber) представляют собой легковесные потоки, которые требуют от приложения управления их переключением. Они работают в контексте потоков, но имеют собственные стек и состояние.

Нить **не является «полноценным» объектом ядра**. Идентификация нитей – указатель void* (LPVOID), который не является дескриптором. Также нити получают локальную память – **Fiber Local Storage (FLS),** подобную TLS.

Создание, завершение, переключение нитей

- Создание нитей: выполняется с помощью API, который выделяет необходимый стек и устанавливает контекст исполнения.
- Завершение нитей: происходит, когда выполнение нити завершается, и ресурсы освобождаются.
- Переключение нитей: осуществляется вручную в коде приложения, что позволяет изменять активную нить без вмешательства ОС, осуществляется в пределах одного потока.

Взаимосвязь нитей и потоков

- Нити выполняются внутри потоков и могут быть переключены без участия операционной системы, что делает их более эффективными для задач, требующих частого переключения контекста.
- Потоки управляются операционной системой, а нити — программным обеспечением на уровне приложения.

Задание, Job – группировка и совместное управление несколькими процессами. Задание – системный объект, идентифицируемый глобальным именем (необязательным) и дескриптором (handle) в пространстве дескрипторов процесса.

Приоритет, Priority – значение (обычно целочисленное, от нескольких единиц до нескольких десятков), используемое системным планировщиком и характеризующее важность и срочность выполнения процесса.

Разновидности приоритетов:

- Статический приоритет – остается неизменным в течение всего жизненного цикла процесса;
- Динамический приоритет – меняется в зависимости от дополнительных условий;
- Относительный приоритет – выбор и переключение процессов выполняются после завершения текущего процесса, перехода его в состояние ожидания или окончания кванта времени;
- Абсолютный приоритет – при появлении готового к выполнению более приоритетного процесса текущий прерывается и вытесняется немедленно;

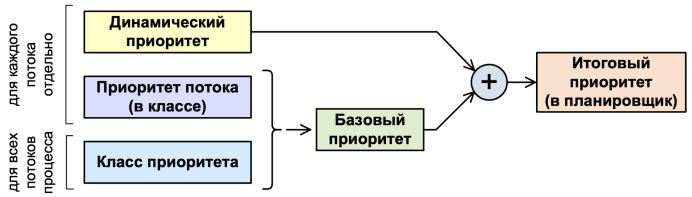
Общее правило: в правильно построенной сбалансированной системе чем выше приоритет, тем меньше общее время исполнения.

Приоритеты в Windows – целочисленные значения в диапазоне от 0 до 31: 0 – зарезервированное значение для специального системного потока **«zero-page»**, и от 1 до 31 – для всех остальных. Большее значение соответствует большему приоритету. Приоритет «обычных» процессов ограничен – от 1 до 15; приоритет 16 и выше могут иметь только привилегированные процессы.

Нулевая страница или базовая страница — это блок памяти в самом начале адресного пространства компьютера

Значение приоритета – составное, оно образуется из трех составляющих: класс приоритета, приоритет потока в классе, динамическое изменение приоритета («boost»).

Класс приоритета, Priority Class (6) – присваивается процессу. Все потоки одного процесса относятся к одному классу.
Приоритет потока или приоритет в классе, Thread Priority (7) – присваивается конкретному потоку.
Класс приоритета и приоритет потока в классе образуют **базовый** приоритет.



Динамический приоритет, Dynamic Priority – действующее значение приоритета с учетом временного его повышения или понижения.
Повышение приоритета (priority boost)

Основные объекты в реализации многозадачности Windows: вычислительный процесс (**process**), вычислительный поток (**thread**). Дополнительно: нить (**fiber**), задание (**job**).

Модель многозадачности и дисциплина планирования – многопоточная, с квантованием времени, вытесняющая с абсолютными приоритетами.

WinAPI команды для работы с:

- Процессами: CreateProcess, ExitProcess, TerminateProcess (Завершение другого процесса «извне» по его дескриптору), OpenProcess (Открыть для доступа по его ID).
- Потоками: CreateThread, CreateRemoteThread (в адресном пространстве другого процесса), ExitThread, TerminateThread (принудительное завершение потока), GetExitCodeThread, SuspendThread (приостановка может быть многократной), ResumeThread.
- Нити: CreateFiber, SwitchToFiber, DeleteFiber.
- Задание: CreateJobObject, OpenJobObject
- Приоритеты: SetPriorityClass, GetThreadPriority, GetProcessPriorityBoost

Виды процессов (программ):

- Оконные (windowed)
- Консольные (console)
- Службы/сервисы (service)
- Системные

Виртуальная память — это механизм операционной системы, который использует диск для расширения доступной оперативной памяти, позволяя работать с большими объемами данных и изолируя процессы. Динамическая память, в свою очередь, управляет выделением и освобождением памяти во время выполнения программы, позволяя эффективно использовать ресурсы в зависимости от потребностей. Виртуальная память работает на уровне ОС, а динамическая — на уровне приложений; первая обеспечивает доступность памяти, вторая — управление ресурсами.