**Поток** — это по сути последовательность инструкций, которые выполняются параллельно с другими потоками. Каждая программа создает по меньшей мере один поток: основной, который запускает функцию main(). Программа, использующая только главный поток, является однопоточной; если добавить один или более потоков, она станет многопоточной.

**потоки — это способ сделать несколько вещей одновременно.**

**GetCurrentProcessId()** - идентификатор вызывающего процесса.

**GetCurrentThreadId()** - идентификатор вызывающего потока Дескрипторы не наследуются дочерними процессами.

**Псевдодескриптор** - это специальная константа, которая интерпретируется как дескриптор текущего потока. Вызывающий поток может использовать этот дескриптор, чтобы указать себя всякий раз, когда требуется дескриптор потока.

**Windows API** — общее наименование набора базовых функций [интерфейсов программирования приложений](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) операционных систем семейств [*Microsoft Windows*](https://ru.wikipedia.org/wiki/Microsoft_Windows)

Предоставляет прямой способ взаимодействия приложений пользователя с операционной системой *Windows*.

**CreateThread()** - Создает поток для выполнения в виртуальном адресном пространстве вызывающего процесса.

***Параметры:***

1 Указатель на структуру [SECURITY\_ATTRIBUTES,](https://docs.microsoft.com/en-us/previous-versions/windows/desktop/legacy/aa379560(v=vs.85)) которая определяет, **может ли возвращаемый дескриптор быть унаследован дочерними процессами**. Если *lpThreadAttributes* имеет значение NULL, дескриптор не может быть унаследован.

2 **Начальный размер стека в байтах**. Система округляет это значение до ближайшей страницы. **Если этот параметр равен нулю, новый поток использует размер исполняемого файла по умолчанию.**

3 **Указатель на функцию, определяемую приложением, которая должна выполняться потоком**. Этот указатель представляет начальный адрес потока.

4 Указатель на переменную, которую нужно передать потоку.

5 Флаги, управляющие созданием потока.

6 Указатель на переменную, которая получает идентификатор потока. Если этот параметр равен NULL , идентификатор потока не возвращается.

**SuspendThread()** -Приостанавливает указанный поток.

***Параметры:***

1 Дескриптор потока, который должен быть приостановлен.

**ResumeThread()** - Уменьшает количество приостановленных потоков. Когда счетчик приостановки уменьшается до нуля, выполнение потока возобновляется.

***Параметры:***

1 Дескриптор перезапускаемого потока.

**TerminateThread()** - Завершает поток.

***Параметры:***

1 Дескриптор завершаемого потока.

2 Код выхода для потока.

**WaitForSingleObject** останавливает выполнения программы до тех пор пока объект , идентификатор которого передан в функцию ,не окажется в "сигнальном" состоянии

**CloseHandle** аннулирует заданный дескриптор объекта, уменьшает итоговое число дескрипторов объекта и выполняет проверку наличия объекта. После того, как последний дескриптор объект закрывается, объект удаляется из системы.

В начале создается потоковая функция. Затем новый поток создается функцией pthread\_create()

Функция pthread\_join() блокирует вызывающий поток, пока указанный поток не завершится. Указанный поток должен принадлежать текущему процессу и не должен быть отделен.

| int pthread\_create(\*ptherad\_t, const pthread\_attr\_t \*attr, void\* (\*start\_routine)(void\*), void \*arg); |
| --- |

Функция получает в качестве аргументов указатель на поток, переменную типа pthread\_t, в которую, в случае удачного завершения сохраняет id потока. pthread\_attr\_t – атрибуты потока. В случае если используются атрибуты по умолчанию, то можно передавать NULL. start\_routin – это непосредственно та функция, которая будет выполняться в новом потоке. arg – это аргументы, которые будут переданы функции.

| int pthread\_join(pthread\_t thread, void \*\*value\_ptr); |
| --- |



Приводит к тому, что основной поток будет ждать завершения порожденного. Откладывает выполнение вызывающего (эту функцию) потока, до тех пор, пока не будет выполнен поток thread. Когда pthread\_join выполнилась успешно, то она возвращает 0. Если поток явно вернул значение (это то самое значение SUCCESS, из нашей функции), то оно будет помещено в переменную value\_ptr.

**Легковесный процесс** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *light-weight process*, LWP) — является средством достижения [многозадачности](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C) в компьютерной [операционной системе](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), в традиционном понимании этого термина. В [Unix System V](https://ru.wikipedia.org/wiki/Unix_System_V) и [Solaris](https://ru.wikipedia.org/wiki/Solaris), легковесный процесс работает в [пространстве пользователя](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%BE_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) поверх одного [потока выполнения](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F) ядра, разделяет виртуальное адресное пространство и системные ресурсы потока выполнения с другими легковесными процессами, в рамках того же процесса. Несколько потоков пользовательского уровня, управляемые с помощью библиотеки потоков, могут быть размещены в одном или нескольких легковесных процессах, что даёт многозадачность на уровне пользователя, которая может иметь некоторые преимущества в производительности.