**OpenCL** (**Open Computing Language**) is a [framework](https://en.wikipedia.org/wiki/Software_framework) for writing programs that execute across [heterogeneous](https://en.wikipedia.org/wiki/Heterogeneous_computing) platforms consisting of [central processing units](https://en.wikipedia.org/wiki/Central_processing_unit) (CPUs), [graphics processing units](https://en.wikipedia.org/wiki/Graphics_processing_unit) (GPUs), [digital signal processors](https://en.wikipedia.org/wiki/Digital_signal_processor) (DSPs), [field-programmable gate arrays](https://en.wikipedia.org/wiki/Field-programmable_gate_array) (FPGAs) and other processors or [hardware accelerators](https://en.wikipedia.org/wiki/Hardware_accelerator). OpenCL specifies [programming languages](https://en.wikipedia.org/wiki/Programming_language) (based on [C99](https://en.wikipedia.org/wiki/C99) and [C++11](https://en.wikipedia.org/wiki/C%2B%2B11)) for programming these [devices](https://en.wikipedia.org/wiki/Personal_computer_hardware) and [application programming interfaces](https://en.wikipedia.org/wiki/Application_programming_interface) (APIs) to control the platform and execute programs on the [compute devices](https://en.wikipedia.org/wiki/OpenCL_compute_devices). OpenCL provides a standard interface for [parallel computing](https://en.wikipedia.org/wiki/Parallel_computing) using [task-](https://en.wikipedia.org/wiki/Task_parallelism) and [data-based parallelism](https://en.wikipedia.org/wiki/Data_parallelism).

OpenCL is an open standard maintained by the [non-profit](https://en.wikipedia.org/wiki/Non-profit_organization) technology consortium [Khronos Group](https://en.wikipedia.org/wiki/Khronos_Group" \o "Khronos Group). Conformant implementations are available from [Altera](https://en.wikipedia.org/wiki/Altera), [AMD](https://en.wikipedia.org/wiki/Advanced_Micro_Devices), [Apple](https://en.wikipedia.org/wiki/Apple_Inc) (OpenCL along with [OpenGL](https://en.wikipedia.org/wiki/OpenGL) is [deprecated](https://en.wikipedia.org/wiki/OpenGL#Deprecation_in_Apple_devices) for Apple hardware, in favor of [Metal](https://en.wikipedia.org/wiki/Metal_(API)) 2[[7]](https://en.wikipedia.org/wiki/OpenCL#cite_note-7)), [ARM](https://en.wikipedia.org/wiki/ARM_Holdings), [Creative](https://en.wikipedia.org/wiki/Creative_Technology), [IBM](https://en.wikipedia.org/wiki/IBM), [Imagination](https://en.wikipedia.org/wiki/Imagination_Technologies), [Intel](https://en.wikipedia.org/wiki/Intel), [Nvidia](https://en.wikipedia.org/wiki/Nvidia), [Qualcomm](https://en.wikipedia.org/wiki/Qualcomm), [Samsung](https://en.wikipedia.org/wiki/Samsung), [Vivante](https://en.wikipedia.org/wiki/Vivante), [Xilinx](https://en.wikipedia.org/wiki/Xilinx), and [ZiiLABS](https://en.wikipedia.org/wiki/ZiiLABS" \o "ZiiLABS).[[8]](https://en.wikipedia.org/wiki/OpenCL#cite_note-8)[[9]](https://en.wikipedia.org/wiki/OpenCL#cite_note-9)

## Overview[[edit](https://en.wikipedia.org/w/index.php?title=OpenCL&action=edit&section=1)]

OpenCL views a computing system as consisting of a number of *compute devices*, which might be [central processing units](https://en.wikipedia.org/wiki/Central_processing_unit) (CPUs) or "accelerators" such as graphics processing units (GPUs), attached to a *host* processor (a CPU). It defines a [C-like language](https://en.wikipedia.org/wiki/OpenCL#OpenCL_C_language) for writing programs. Functions executed on an OpenCL device are called "[kernels](https://en.wikipedia.org/wiki/Compute_kernel)".[[10]](https://en.wikipedia.org/wiki/OpenCL#cite_note-specification-10):17 A single compute device typically consists of several *compute units*, which in turn comprise multiple [*processing elements*](https://en.wikipedia.org/wiki/Processing_element) (PEs). A single kernel execution can run on all or many of the PEs in parallel. How a compute device is subdivided into compute units and PEs is up to the vendor; a compute unit can be thought of as a "[core](https://en.wikipedia.org/wiki/Processor_core)", but the notion of core is hard to define across all the types of devices supported by OpenCL (or even within the category of "CPUs"),[[11]](https://en.wikipedia.org/wiki/OpenCL#cite_note-Gaster-11):49–50 and the number of compute units may not correspond to the number of cores claimed in vendors' marketing literature (which may actually be counting [SIMD lanes](https://en.wikipedia.org/wiki/SIMD_lanes)).[[12]](https://en.wikipedia.org/wiki/OpenCL#cite_note-12)

In addition to its C-like programming language, OpenCL defines an [application programming interface](https://en.wikipedia.org/wiki/Application_programming_interface) (API) that allows programs running on the host to launch kernels on the compute devices and manage device memory, which is (at least conceptually) separate from host memory. Programs in the OpenCL language are intended to be [compiled at run-time](https://en.wikipedia.org/wiki/Just-in-time_compilation), so that OpenCL-using applications are portable between implementations for various host devices.[[13]](https://en.wikipedia.org/wiki/OpenCL#cite_note-CiSE-13) The OpenCL standard defines host APIs for [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B); third-party APIs exist for other programming languages and platforms such as [Python](https://en.wikipedia.org/wiki/Python_(programming_language)),[[14]](https://en.wikipedia.org/wiki/OpenCL#cite_note-pyopencl-14) [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [Perl](https://en.wikipedia.org/wiki/Perl)[[15]](https://en.wikipedia.org/wiki/OpenCL#cite_note-perl-opencl-15) and [.NET](https://en.wikipedia.org/wiki/.NET_Framework).[[11]](https://en.wikipedia.org/wiki/OpenCL#cite_note-Gaster-11):15 An [implementation](https://en.wikipedia.org/wiki/OpenCL#Implementations) of the OpenCL standard consists of a [library](https://en.wikipedia.org/wiki/Library_(computing)) that implements the API for C and C++, and an OpenCL C [compiler](https://en.wikipedia.org/wiki/Compiler) for the compute device(s) targeted.

In order to open the OpenCL programming model to other languages or to protect the kernel source from inspection, the [Standard Portable Intermediate Representation](https://en.wikipedia.org/wiki/Standard_Portable_Intermediate_Representation) (SPIR)[[16]](https://en.wikipedia.org/wiki/OpenCL#cite_note-16) can be used as a target-independent way to ship kernels between a front-end compiler and the OpenCL back-end.

More recently [Khronos Group](https://en.wikipedia.org/wiki/Khronos_Group" \o "Khronos Group) has ratified [SYCL](https://en.wikipedia.org/wiki/SYCL),[[17]](https://en.wikipedia.org/wiki/OpenCL#cite_note-17) a higher-level programming model for OpenCL as single-source [DSEL](https://en.wikipedia.org/wiki/DSEL) based on pure [C++11](https://en.wikipedia.org/wiki/C%2B%2B11) to improve [programming productivity](https://en.wikipedia.org/wiki/Programming_productivity).

### Memory hierarchy[[edit](https://en.wikipedia.org/w/index.php?title=OpenCL&action=edit&section=2)]

OpenCL defines a four-level [memory hierarchy](https://en.wikipedia.org/wiki/Memory_hierarchy) for the compute device:[[13]](https://en.wikipedia.org/wiki/OpenCL#cite_note-CiSE-13)

* global memory: shared by all processing elements, but has high access latency (\_\_global);
* read-only memory: smaller, low latency, writable by the host CPU but not the compute devices (\_\_constant);
* local memory: shared by a group of processing elements (\_\_local);
* per-element private memory ([registers](https://en.wikipedia.org/wiki/Processor_register); \_\_private).

Not every device needs to implement each level of this hierarchy in hardware. [Consistency](https://en.wikipedia.org/wiki/Consistency_model) between the various levels in the hierarchy is relaxed, and only enforced by explicit [synchronization](https://en.wikipedia.org/wiki/Synchronization_(computer_science)) constructs, notably [barriers](https://en.wikipedia.org/wiki/Memory_barrier).

Devices may or may not share memory with the host CPU.[[13]](https://en.wikipedia.org/wiki/OpenCL#cite_note-CiSE-13) The host API provides [handles](https://en.wikipedia.org/wiki/Handle_(computing)) on device memory buffers and functions to transfer data back and forth between host and devices.

## OpenCL C language[[edit](https://en.wikipedia.org/w/index.php?title=OpenCL&action=edit&section=3)]

The programming language that is used to write [compute kernels](https://en.wikipedia.org/wiki/Compute_kernel) is called OpenCL C and is based on [C99](https://en.wikipedia.org/wiki/C99),[[18]](https://en.wikipedia.org/wiki/OpenCL#cite_note-openclc-18) but adapted to fit the device model in OpenCL. Memory buffers reside in specific levels of the [memory hierarchy](https://en.wikipedia.org/wiki/OpenCL#Memory_hierarchy), and [pointers](https://en.wikipedia.org/wiki/Pointer_(computer_programming)) are annotated with the region qualifiers \_\_global, \_\_local, \_\_constant, and \_\_private, reflecting this. Instead of a device program having a main function, OpenCL C functions are marked \_\_kernel to signal that they are [entry points](https://en.wikipedia.org/wiki/Entry_point) into the program to be called from the host program. [Function pointers](https://en.wikipedia.org/wiki/Function_pointer), [bit fields](https://en.wikipedia.org/wiki/Bit_field) and [variable-length arrays](https://en.wikipedia.org/wiki/Variable-length_array) are omitted, and [recursion](https://en.wikipedia.org/wiki/Recursion_(computer_science)) is forbidden.[[19]](https://en.wikipedia.org/wiki/OpenCL#cite_note-AMD-19) The [C standard library](https://en.wikipedia.org/wiki/C_standard_library) is replaced by a custom set of standard functions, geared toward math programming.

OpenCL C is extended to facilitate use of [parallelism](https://en.wikipedia.org/wiki/Parallel_computing) with vector types and operations, synchronization, and functions to work with work-items and work-groups.[[19]](https://en.wikipedia.org/wiki/OpenCL#cite_note-AMD-19) In particular, besides scalar types such as float and double, which behave similarly to the corresponding types in C, OpenCL provides fixed-length vector types such as float4 (4-vector of single-precision floats); such vector types are available in lengths two, three, four, eight and sixteen for various base types.[[18]](https://en.wikipedia.org/wiki/OpenCL#cite_note-openclc-18):§ 6.1.2 [Vectorized](https://en.wikipedia.org/wiki/Array_programming) operations on these types are intended to map onto [SIMD](https://en.wikipedia.org/wiki/SIMD) instructions sets, e.g., [SSE](https://en.wikipedia.org/wiki/Streaming_SIMD_Extensions) or [VMX](https://en.wikipedia.org/wiki/AltiVec), when running OpenCL programs on CPUs.[[13]](https://en.wikipedia.org/wiki/OpenCL#cite_note-CiSE-13) Other specialized types include 2-d and 3-d image types.

OpenCL was initially developed by [Apple Inc.](https://en.wikipedia.org/wiki/Apple_Inc.), which holds [trademark](https://en.wikipedia.org/wiki/Trademark) rights, and refined into an initial proposal in collaboration with technical teams at [AMD](https://en.wikipedia.org/wiki/Advanced_Micro_Devices), [IBM](https://en.wikipedia.org/wiki/IBM), [Qualcomm](https://en.wikipedia.org/wiki/Qualcomm), [Intel](https://en.wikipedia.org/wiki/Intel_Corporation), and [Nvidia](https://en.wikipedia.org/wiki/Nvidia). Apple submitted this initial proposal to the [Khronos Group](https://en.wikipedia.org/wiki/Khronos_Group" \o "Khronos Group). On June 16, 2008, the Khronos Compute Working Group was formed[[23]](https://en.wikipedia.org/wiki/OpenCL#cite_note-23) with representatives from CPU, GPU, embedded-processor, and software companies. This group worked for five months to finish the technical details of the specification for OpenCL 1.0 by November 18, 2008.[[24]](https://en.wikipedia.org/wiki/OpenCL#cite_note-macWorld-24) This technical specification was reviewed by the Khronos members and approved for public release on December 8, 2008.

### OpenCL 1.0[[edit](https://en.wikipedia.org/w/index.php?title=OpenCL&action=edit&section=7)]

OpenCL 1.0 released with [Mac OS X Snow Leopard](https://en.wikipedia.org/wiki/Mac_OS_X_Snow_Leopard) on August 28, 2009. According to an Apple press release:[[26]](https://en.wikipedia.org/wiki/OpenCL#cite_note-pressrelease-26)

Snow Leopard further extends support for modern hardware with Open Computing Language (OpenCL), which lets any application tap into the vast gigaflops of GPU computing power previously available only to graphics applications. OpenCL is based on the C programming language and has been proposed as an open standard.

AMD decided to support OpenCL instead of the now deprecated [Close to Metal](https://en.wikipedia.org/wiki/Close_to_Metal) in its [Stream framework](https://en.wikipedia.org/wiki/AMD_Stream_SDK).[[27]](https://en.wikipedia.org/wiki/OpenCL#cite_note-AMDpressrelease-27)[[28]](https://en.wikipedia.org/wiki/OpenCL#cite_note-eweekAMD-28) [RapidMind](https://en.wikipedia.org/wiki/RapidMind" \o "RapidMind) announced their adoption of OpenCL underneath their development platform to support GPUs from multiple vendors with one interface.[[29]](https://en.wikipedia.org/wiki/OpenCL#cite_note-RapidMindHPCWire-29) On December 9, 2008, Nvidia announced its intention to add full support for the OpenCL 1.0 specification to its GPU Computing Toolkit.[[30]](https://en.wikipedia.org/wiki/OpenCL#cite_note-Nvidia_Press_Release_2008-12-09-30) On October 30, 2009, IBM released its first OpenCL implementation as a part of the [XL compilers](https://en.wikipedia.org/w/index.php?title=XL_compilers&action=edit&redlink=1).[[31]](https://en.wikipedia.org/wiki/OpenCL#cite_note-openclIBM-31)

### OpenCL 3.0[[edit](https://en.wikipedia.org/w/index.php?title=OpenCL&action=edit&section=13)]

OpenCL 3.0 is in Final Mode (2020-09-30). OpenCL 1.2 is mandatory. All OpenCL 2.x Modules and new 3.0 modules are optional. New preferred language is C++ for OpenCL with many C++17 features. [[44]](https://en.wikipedia.org/wiki/OpenCL#cite_note-44)[[45]](https://en.wikipedia.org/wiki/OpenCL#cite_note-45)[[46]](https://en.wikipedia.org/wiki/OpenCL#cite_note-46)

When releasing OpenCL 2.2, the Khronos Group announced that OpenCL would converge where possible with [Vulkan](https://en.wikipedia.org/wiki/Vulkan_(API)) to enable OpenCL software deployment flexibility over both APIs.[[47]](https://en.wikipedia.org/wiki/OpenCL#cite_note-47)[[48]](https://en.wikipedia.org/wiki/OpenCL#cite_note-48) This has been now demonstrated by Adobe's Premiere Rush using the clspv[[49]](https://en.wikipedia.org/wiki/OpenCL#cite_note-49) open source compiler to compile significant amounts of OpenCL C kernel code to run on a Vulkan runtime for deployment on Android.[[50]](https://en.wikipedia.org/wiki/OpenCL#cite_note-50) OpenCL has a forward looking roadmap independent of Vulkan, with 'OpenCL Next' under development and targeting release in 2020. OpenCL Next may integrate extensions such as Vulkan / OpenCL Interop, Scratch-Pad Memory Management, Extended Subgroups, SPIR-V 1.4 ingestion and SPIR-V Extended debug info. OpenCL is also considering Vulkan-like loader and layers and a ‘Flexible Profile’ for deployment flexibility on multiple accelerator types