NAME

clinfo − show OpenCL platforms and devices

SYNOPSIS

clinfo [*options ...*]

DESCRIPTION

clinfo prints all available information about all OpenCL platforms available on the system and the devices they expose.

OPTIONS

clinfo accepts the following options:

-a, **--all-props**
try to retrieve all properties, even those not officially supported (e.g. because they require specific extensions), but only show them if the property could be retrieved successfully; see also the **LIMITATIONS** section below; note that even though this may reveal hidden properties, there is no guarantee that the returned values are meaningful, nor that the corresponding feature is actually available at all;

-A, **--always-all-props**
like **-a**, but also show errors;

--human
produce human-friendly output; this is the default (except as noted below);

--raw
produce machine-friendly output; this is the default if **clinfo** is invoked with a name that contains the string "*raw*";

--json
outputs the raw data (cf. the **--raw** option) in JSON format; support for this option is experimental, as the representation of some of the values is not finalized;

--offline
shows also offline devices for platforms that expose this feature;

--null-platform
tries to handle the NULL platform as a normal platform, retrieving and showing its properties and devices; this is in addition to the NULL platform behavior tests done at the end, and can be useful on systems where there are no ICD platforms, but there is a platform hard-coded in the OpenCL library itself;

-l, **--list**
list platforms and devices by name, with no (other) properties;

-d *platform_index*:*device_index*
--device *platform_index*:*device_index*
only show properties for the specified device in the specified platform; multiple device specifications may be given on the command-line;

--prop *property-name*
only show properties whose symbolic name matches (contains as a substring) the given *property-name*; the name is normalized as upper-case and with minus sign (-) replaced by underscore signs (_); multiple property specifications may be given on the command-line; when this flag is specified, raw mode is forced;

--help, **-?**, **-h**
show usage;

--version, **-v**
show program version.

CONFORMING TO

OpenCL 1.1, OpenCL 1.2, OpenCL 2.0, OpenCL 2.1, OpenCL 2.2, OpenCL 3.0.

EXTENSIONS

Supported OpenCL extensions:

cl_khr_device_uuid
for the UUID, LUID and node mask of the device;

cl_khr_extended_versioning
for the extended platform, device, extension and IL versioned properties backported from OpenCL 3.0 to previous OpenCL versions;

cl_khr_fp16, **cl_khr_fp64**, **cl_amd_fp64**, **cl_APPLE_fp64_basic_ops**
for information about support for half-precision and double-precision floating-point data types;

cl_khr_image2d_from_buffer
for information about the base address and pitch alignment requirements of buffers to be used as base for 2D images;

cl_khr_il_program
for information about the supported IL (Intermediate Language) representations;

cl_khr_spir
for information about the supported SPIR (Standard Portable Intermediate Representation) versions;

cl_khr_icd
for the suffix of vendor extensions functions;

cl_khr_subgroup_named_barrier
for the maximum number of named sub-group barriers;

cl_khr_terminate_context, **cl_arm_controlled_kernel_termination**
for the terminate capabilities for the device;

cl_ext_device_fission
for device fission support in OpenCL 1.1 devices;

cl_khr_pci_bus_info
for the PCI bus information (see also **cl_nv_device_attribute_query** and **cl_amd_device_attribute_query**)

cl_ext_atomic_counters_32
cl_ext_atomic_counters_64
for the atomic counter extension;

cl_ext_float_atomics
for the floating-point atomic capabilities for half, single and double precision (depending on hardware floating-point size support);

cl_ext_cxx_for_opencl
for the version of the C++ for OpenCL language supported by the device compiler;

cl_amd_device_attribute_query
for AMD-specific device attributes;

cl_amd_object_metadata
to show the maximum number of keys supported by the platform;

cl_amd_offline_devices
to show offline devices exposed by the platform, if requested (see **--offline** option);

cl_amd_copy_buffer_p2p
to show the number and IDs of available P2P devices;

cl_amd_svm
cl_arm_shared_virtual_memory
for Shared Virtual Memory (SVM) capabilities in OpenCL 1.2 devices;

cl_arm_core_id
to show the (potentially sparse) list of the core IDs that the device may return;

cl_arm_job_slot_selection
to show the (potentially sparse) list of available job slots for command submission;

cl_arm_scheduling_controls
to show the supported work scheduling controls and the available sets of register allocations;

cl_nv_device_attribute_query
for NVIDIA-specific device attributes;

cl_intel_device_attribute_query
for Intel-specific device attributes;

cl_intel_exec_by_local_thread
for the Intel extension allowing CPU devices to run kernels as part of the current host thread;

cl_intel_advanced_motion_estimation
for the version of the Intel Motion Estimation accelerator version;

cl_intel_device_side_avc_motion_estimation
for the version and supported features of Intel's device-side AVC Motion;

cl_intel_planar_yuv
for the maximum dimensions of planar YUV images;

cl_intel_simultaneous_sharing
for simultaneous CL/GL/DirectX context sharing (only partial support);

cl_intel_required_subgroup_size
to enumerate allowed sub-group sizes;

cl_intel_command_queue_families
to enumerate the available command queues and their properties and capabilities;

cl_altera_device_temperature
for the Altera extension to query the core temperature of the device;

cl_qcom_ext_host_ptr
for the QUALCOMM extension to query page size and required padding in external memory allocation.

NOTES

Some information is duplicated when available from multiple sources. Examples:

- supported device partition types and domains as obtained using the **cl_ext_device_fission** extension typically match the ones obtained using the core OpenCL 1.2 device partition feature;
- the preferred work-group size multiple matches the NVIDIA warp size (on NVIDIA devices) or the AMD wavefront width (on AMD devices).

Some floating-point configuration flags may only be meaningful for specific precisions and/or specific OpenCL versions. For example, **CL_FP_CORRECTLY_ROUNDED_DIVIDE_SQRT** is only relevant for single precision in OpenCL 1.2 devices.

The implementation-defined behavior for NULL platform or context properties is tested for the following API calls:

clGetPlatformInfo()
by trying to show the platform name;

clGetDeviceIDs()
by trying to enumerate devices; the corresponding platform (if any) is then detected by querying the device platform of the first device;

clCreateteContext()
by trying to create a context from a device from the previous list (if any), and a context from a device from a different platform;

clCreateteContextFromType()
by trying to create contexts for each device type (except DEFAULT).

EXPERIMENTAL FEATURES

Support for OpenCL 2.x properties is not fully tested.

Support for **cl_khr_subgroup_named_barrier** is experimental due to missing definitions in the official OpenCL headers.

Raw (machine-parsable) output is considered experimental, the output format might still undergo changes.

The properties of the ICD loader will also be queried if the **clGetICDLoaderInfoOCLICD** extension function is found.

Support for the properties exposed by **cl_amd_copy_buffer_p2p** is experimental.

Support for some (documented and undocumented) properties exposed by **cl_amd_device_attribute_query** is experimental (see also **LIMITATIONS**).

Support for the interop lists exposed by **cl_intel_simultaneous_sharing** is experimental.

The highest OpenCL version supported by the ICD loader is detected with some trivial heuristics (symbols found); a notice is output if this is lower than the highest platform OpenCL version, or if the detected version doesn't match the one declared by the ICD loader itself.

LIMITATIONS

OpenCL did not provide an explicit mean to detect the supported version of any extension exposed by a device until version 3.0. This makes it impossible in many circumstances to determine a priori if it will be possible to successfully query a device about a specific property even if it declares support for a given extension. Additionally, the actual size and meaning of some properties are not officially declared anywhere.

Most notably, this affects extensions such as **cl_amd_device_attribute_query**, **cl_nv_device_attribute_query** and **cl_arm_core_id**. Heuristics based on standard version support are partially used in the code to determine which version may be supported.

Properties which are known to be affected by these limitations include:

CL_DEVICE_GLOBAL_FREE_MEMORY_AMD
documented in v3 of the **cl_amd_device_attribute_query** extension specification as being the global free memory in KBytes, without any explanation given on why there are two values, although in the source code of the **ROCm** stack the second value is documented as being the largest free block;

CL_DEVICE_AVAILABLE_ASYNC_QUEUES_AMD
documented in v3 of the **cl_amd_device_attribute_query** extension specification, but not reported by drivers supporting other v3 properties. This has now been enabled for drivers *assumed* to support v4 of the same extension;

CL_DEVICE_TERMINATE_CAPABILITY_KHR
exposed by the **cl_khr_terminate_context** has changed value between OpenCL 1.x and 2.x, and it's *allegedly* a bitfield, whose values are however not defined anywhere.

BUGS

General

Please report any issues on the project tracker on GitHub ⟨http://github.com/Oblomov/clinfo⟩.

LLVM CommandLine errors

If multiple OpenCL platforms using shared **LLVM** libraries are present in the system, **clinfo** (and other OpenCL application) may crash with errors to the tune of

: CommandLine Error: Option '(some option name)' registered more than once!
LLVM ERROR: inconsistency in registered CommandLine options

or similar. This is not an issue in **clinfo**, or in any OpenCL platform or application, but it is due to the way **LLVM** handles its own command-line options parsing. The issue has been reported upstream as issue #30587 ⟨https://bugs.llvm.org/show_bug.cgi?id=30587⟩. See the next point for possible workarounds and assistance in identifying the conflicting platforms.

Segmentation faults

Faulty OpenCL platforms may cause segmentation faults in **clinfo** during the information gathering phase, sometimes even before any output is shown. There is very little **clinfo** can do to avoid this. If you see this happening, try disabling all platforms and then re-enabling them one by one until you experience the crash again. Chances are the last platform you enabled is defective in some way (either by being incompatible with other platforms or by missing necessary components and not handling their absence gracefully).

To selectively enable/disable platforms, one way is to move or rename the*.*icd* files present in */etc/OpenCL/vendors/* and then restoring them one by one. When using the free-software **ocl-icd** OpenCL library, a similar effect can be achieved by setting the **OPENCL_VENDOR_PATH** or **OCL_ICD_VENDORS** environment variables, as documented in **libOpenCL**(7). Other implementations of**libOpenCL** are known to support **OPENCL_VENDOR_PATH** too.

Example

find /etc/OpenCL/vendors/ -name '*.icd' | while read OPENCL_VENDOR_PATH ; do clinfo -l > /dev/null ; echo "$? ${OPENCL_VENDOR_PATH}" ; done

This one liner will run **clinfo -l** for each platform individually (hiding the normal output), and report the *.icd* path prefixed by **0** for successful runs, and a non-zero value for faulty platforms.